НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

Завідувач кафедри

_____ О.В. Коваль
(підпис) (ініціали, прізвище)

“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 “ Програмна інженерія “

на тему “Онтологічний підхід до проектування реєстру інформаційних ресурсів“

Виконав (-ла): студент (-ка) ___4___курсу, групи_ТВ-51_

Форсюк Надія Анатоліївна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник старший викладач Дацюк О. А. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 6.050103 “ Програмна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)

”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту

_____
(прізвище, ім’я, по батькові)

1. Тема роботи _____“Онтологічний підхід до проектування реєстру_____
_____інформаційних ресурсів“_____
_____

керівник роботи_____Дацюк Оксана Антонівна, старший викладач
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”____ 201__р. № __

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи _____Форма реалізації – десктопний-застосунок з інтерфейсом для користувача (WindowsForms). Середовища розробки програмного продукту – MySQl, MySQL WorkBench, Visual Studio._____

4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)____Розглянути можливість побудови реєстру інформаційних ресурсів з використанням онтології. Побудувати алгоритм розгортання структури онтології з мета бази даних. Розробити програмний продукт, який буде відображати онтологію з реляційної моделі до OWL-файлу._____
_____
_____
_____

5. Перелік ілюстративного матеріалу

___«Мета розробки системи для побудови структури онтології інформаційного ресурсу по заданому метаопису реляційного типу», «Функціональна модель системи», «Існуючі програмні рішення», «Модель метабази даних реєстру інформаційних ресурсів системи», «Програмні засоби реалізації», «Структура програмного застосунку», «Алгоритм відображення онтології з реляційної моделі до OWL-файлу», «Приклади інтерфейсу користувача», «Висновки»_____

_____

_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”_10_” ___жовтня_____2018р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	01.12.2018-31.03.2019	
2	Розробка архітектури та загальної структури системи	01-21.04.2019	
3.	Розробка структур окремих підсистем	22-28.04.2019	
4.	Програмна реалізація системи	29.04-13.05.2019	
5.	Оформлення пояснювальної записки	06.05-01.06.2019	
6.	Захист програмного продукту	15.05.2019	
7.	Передзахист	29.05.2019	
8.	Захист	17.06.2019	

Студент _____ Форсюк Н. А.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Дацюк О. А.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою дипломної роботи вивчення можливих підходів до побудови та роботи з реєстром інформаційних ресурсів, використовуючи онтологічний підхід, а також побудова мета бази даних реєстру інформаційних ресурсів та створення системи, яка дозволить будувати структуру онтології інформаційного ресурсу по заданому метаопису реляційного типу. Було проведено огляд існуючих програмних рішень для конвертації даних між OWL та SQL структурами.

Загальний обсяг роботи: 52 сторінок, 26 ілюстрацій, 14 бібліографічних посилань та 3 додатки.

Ключові слова: онтологія, інформаційні ресурси, реєстр інформаційних ресурсів, OWL, метаопис даних.

ABSTRACT

The aim of the work is the examination of possible approaches to the construction and work of the register of information resources using an ontological approach and creation of relational database of the register of information resources and system for building the ontology′s structure based on meta description. The existing programs for conversion the data from OWL format to SQL format and vice versa were analyzed as well as their advantages and disadvantages.

Total volume of the paper: 52 pages, 26 illustrations, 14 bibliography links and 3 appendixes.

Keywords: ontology, informational resources, register of informational resources, OWL, relational databases, meta description.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

1. ІР — інформаційні ресурси.
2. РІР — реєстр інформаційних ресурсів.
3. Метадані — це дані з більш загальної формальної системи, що описує задану систему даних.

ЗМІСТ

ВСТУП

Процес накопичення знань для їх подальшого використання завжди був одним з основних видів людської діяльності. Проблемі визначення ролі інформації і специфіці використовуваних ресурсів, можливості їх концентрації, використання і синтезування в даний час приділяється вирішальне значення.

На даний момент основним фактором, який визначає розвиток суспільства, стають накопичені людством знання і уміння, їх доступність широкому колу користувачів, тому інформаційні системи і ресурси все частіше використовуються у всіх сферах людської діяльності.

Інтелектуалізація виробництв якісно підвищила роль інтелекту, знань та інформації. Таким чином інформація, що представляє собою ключовий фактор розвитку будь-якого напряму професійної (і не тільки) діяльності індивідів, визначила технології обробки власних ресурсів як визначальний напрям подальшого розвитку.

Інформаційне суспільство особливе значення надає інформації і інформаційним ресурсам. Інформація стала масовим товаром, який сьогодні доступний кожному. Інформаційні ресурси активно перетворюються в електронну форму. Все більша кількість людей обирає напрям створення, зберігання і обробки інформації як власну професію.

Інформаційні ресурси дозволяють сконцентрувати і структурувати індивідуальні знання, що є основою для системи державного управління, системи освіти і науки тощо.

Інформаційні ресурси — найважливіша складова будь-якої функціонуючої системи.

Однак, володіння інформацією перестало бути фактором, що надає будь-якого типу переваги. Тому ключовим фактором у сучасних умовах є вміння користуватися інформацією. Значна кількість розрізнених джерел і сховищ інформації (баз знань, інформаційних ресурсів тощо), характеризуються різноманітними способами

представлення, форматами і мовами опису інформації. Для приведення до загальнозрозумілого вигляду інформації, що представлена різними способами, потрібно залучити методи стандартизації представлення інформації.

Документування — це перетворення інформації в об'єкт, що володіє якостями, необхідними для включення його в обіг. Отриманий як результат документування об'єкт (в ряді випадків метаінформація) повинен володіти якостями, що дозволяють йому вільно переміщатися в інформаційній системі. Властивість доступності досягається через стандартизацію самого об'єкта, а крім того, встановлення режиму доступу та формування метаінформаційних ресурсів, які повинні включатися в інформаційні ресурси інших взаємодіючих суб'єктів.

Документування є необхідною передумовою формування інформаційного ресурсу, тому ця діяльність повинна бути регламентована з позицій необхідного (обов'язкового) і стандартизованого документування.

Не кожна інформація може бути частиною інформаційних ресурсів. Для того щоб стати частиною інформаційних ресурсів, інформація повинна бути відповідним чином організована, класифікована, пошук її повинен бути полегшений.

Основна роль в описі знань відводиться онтологіям, які використовуються при проектуванні баз знань, створення експертних систем і систем підтримки прийняття рішень, розробці середовищ, орієнтованих на спільне використання інформації декількома користувачами, і розробці різних пошукових систем.

В даний час онтології використовуються в електронній комерції [5], [7] як засіб обміну інформацією; в медицині як засоби класифікації [8] і основи для побудови експертних систем [9]; в плануванні і проектуванні як засоби, що пропонують оптимальні рішення для стандартного набору завдань, і інших областях.

В області інженерії знань онтології є засобом формування систем керування знаннями [2].

В області інформаційних систем онтології відіграють важливу роль, об'єднуючи основні компоненти будь-якої інформаційної системи: інформаційні ресурси, інтерфейс користувача і прикладні програми.

Тому метою роботи є вивчення можливих підходів до побудови та роботи реєстру інформаційних ресурсів, використовуючи онтологічний підхід.

1. ЗАДАЧА ВИКОРИСТАННЯ ОНТОЛОГІЧНОГО ПІДХОДУ ДО ПРОЕКТУВАННЯ РЕЄСТРУ ІНФОРМАЦІЙНИХ РЕСУРСІВ

Метою роботи є вивчення можливих підходів до побудови та роботи реєстру інформаційних ресурсів, використовуючи онтологічний підхід.

Інформаційний ресурс – будь-який доступний спосіб відображення інформації (найчастіше файл), до інформаційних ресурсів відносять відповідним чином впорядковані і структуровані інформацію і знання. Реєстр інформаційних ресурсів – це метаопис інформаційних ресурсів.

Метадані — дані про дані — дані, що описують сутності, представлені в інформаційних системах, є характеристиками описуваних сутностей для цілей їх ідентифікації, пошуку, оцінки і управління. Метаданими називається структурована інформація, яка описує, пояснює, вказує місце розташування інформаційного ресурсу [12].

Метадані — це зміст каталогів, довідників, реєстрів тощо, що містять відомості про склад даних, зміст, статус, походження, місцезнаходження, як форматах і формах уявлення, умови доступу, придбання і використання авторських, майнових і суміжних з ними правах на дані та ін.

Одним з ефективних способів управління такими даними є збреження реєстру інформаційних ресурсів у реляційній базі даних [10].

Реляційна БД — це сукупність схем відносин пов'язаних один з одним. У реляційних базах даних вся інформація знаходиться в таблицях, рядках і стовпцях, які називаються записами і полями відповідно. Записи в таблицях не повторюються. Їх унікальність забезпечується первинним ключем, що містить набір полів, які однозначно визначають запис.

Однак, такий підхід має свій недолік — важко відслідкувати класифікацію та ієрархію IP каталогів. Онтологічний підхід надасть можливість відображати дерева каталогів як структуру, з легко прослідковуваною ієрархією.

Отже, онтологія — це концептуалізація предметної області; реєстр інформаційних ресурсів відрізняється для кожної предметної області. Тому реєстр інформаційних ресурсів (метадані, які краще зберігати в РБД) пропонується розгортати у онтологію рис.1.

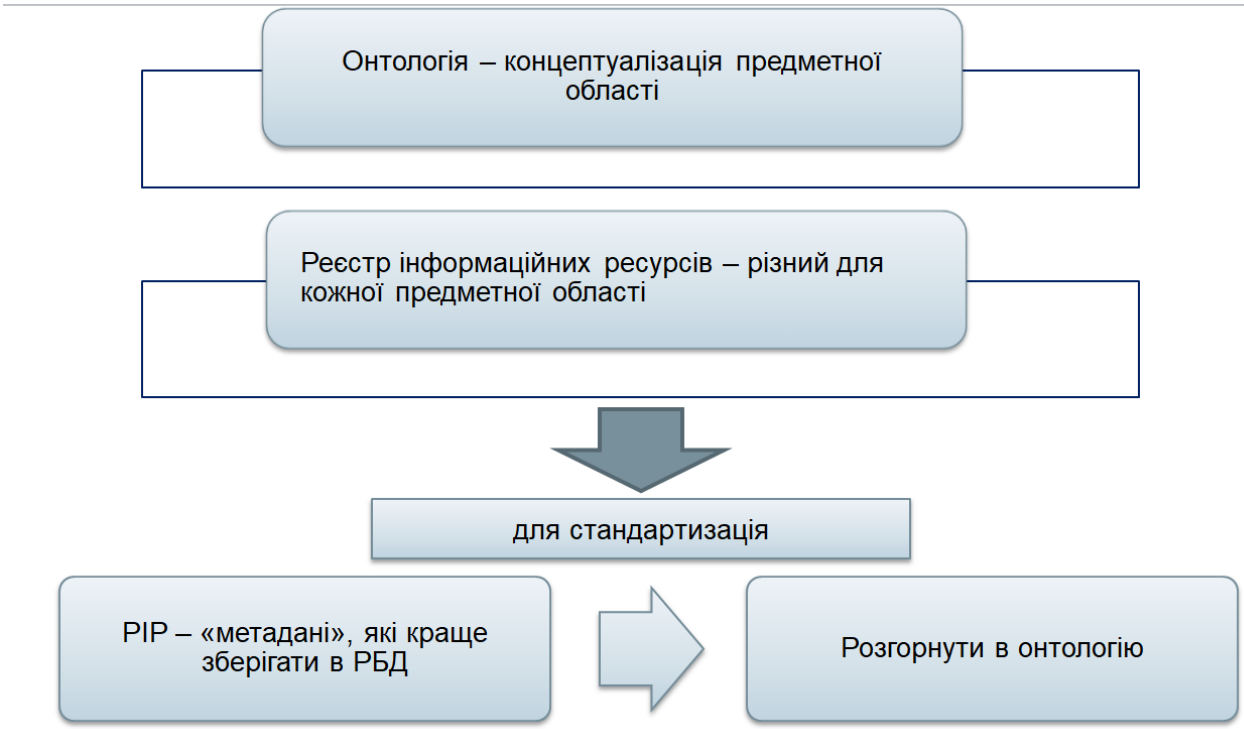

Рисунок 1. – Схема онтологічного підходу побудови PIP

1.1. Задача розробки реляційної бази даних реєстру інформаційних ресурсів

При зростаючому обсязі інформації ми стикаємося з проблемою раціонального управління даними. Оптимальна система пошуку і зберігання даних допомагає не тільки швидше знаходити об'єкти з потрібними атрибутами, а й аналізувати дані. Реалізація такої системи неможлива без явно представлених описів властивостей даних — метаданих. Метадані — це структуровані, кодовані дані, які описують характеристики об'єктів-носіїв інформації, які сприяють ідентифікації, виявлення, оцінки та управління цими об'єктами [4].

Як було сказано, реєстр інформаційних ресурсів — це метаопис інформаційних ресурсів. Метадані зберігають основну інформацію про об'єкт, забезпечують збереження даних і захищають від несанкціонованого доступу [13]. Вони істотно підвищують цінність даних і забезпечують більш широкі можливості їх використання. Зміст метаданих, їх структура, функції і засоби їх уявлення залежать від описуваних ресурсів, предметної області застосованих систем, контексту і характеру їх використання, а також від багатьох інших факторів. Метадані можуть зберігатися разом з об'єктом, що гарантує, що вони не будуть втрачені і при зміні будуть оновлені разом. Однак не завжди можливо вбудувати метадані в деякі типи об'єктів, крім того зберігання метаданих окремо може поліпшити і спростити пошук цих об'єктів. Тому метадані часто зберігаються в базі даних і «пов'язують» з описаними об'єктами [11]. Використання різних СКБД може вимагати різного способу подання даних, а також впливати на легкість керування даними. Тому вибір системи для зберігання метаданих є важливим рішенням.

Для розробки реляційної бази даних реєстру інфомаційних ресурсів було обране середовище MySql — система керування базами даних (СКБД). Подібний клас програмного забезпечення дозволяє структуровано зберігати інформацію, отримувати до неї доступ і управляти безпекою даного доступу за допомогою SQL-запитів.

1.2. Задача розробки системи, яка дозволить будувати структуру онтології інформаційного ресурсу по заданому метаопису реляційного типу

Однією із задач, є задача побудови системи, що дозволяє проектувати структуру онтології інформаційного ресурсу по заданому метаопису реляційного типу. Більшість наявних систем пропонують функціонал, що лише дозволяє відображати інформацію, збережену у базах даних, графічно, у вигляді графів. Такі

системи не підтримують функціоналу для розгортання структури метаданих реляційного типу у вигляді онтології.

На рисунку 2 зображено функціональну модель системи, для розгортання структури онтології інформаційного ресурсу по заданому метаопису реляційного типу.

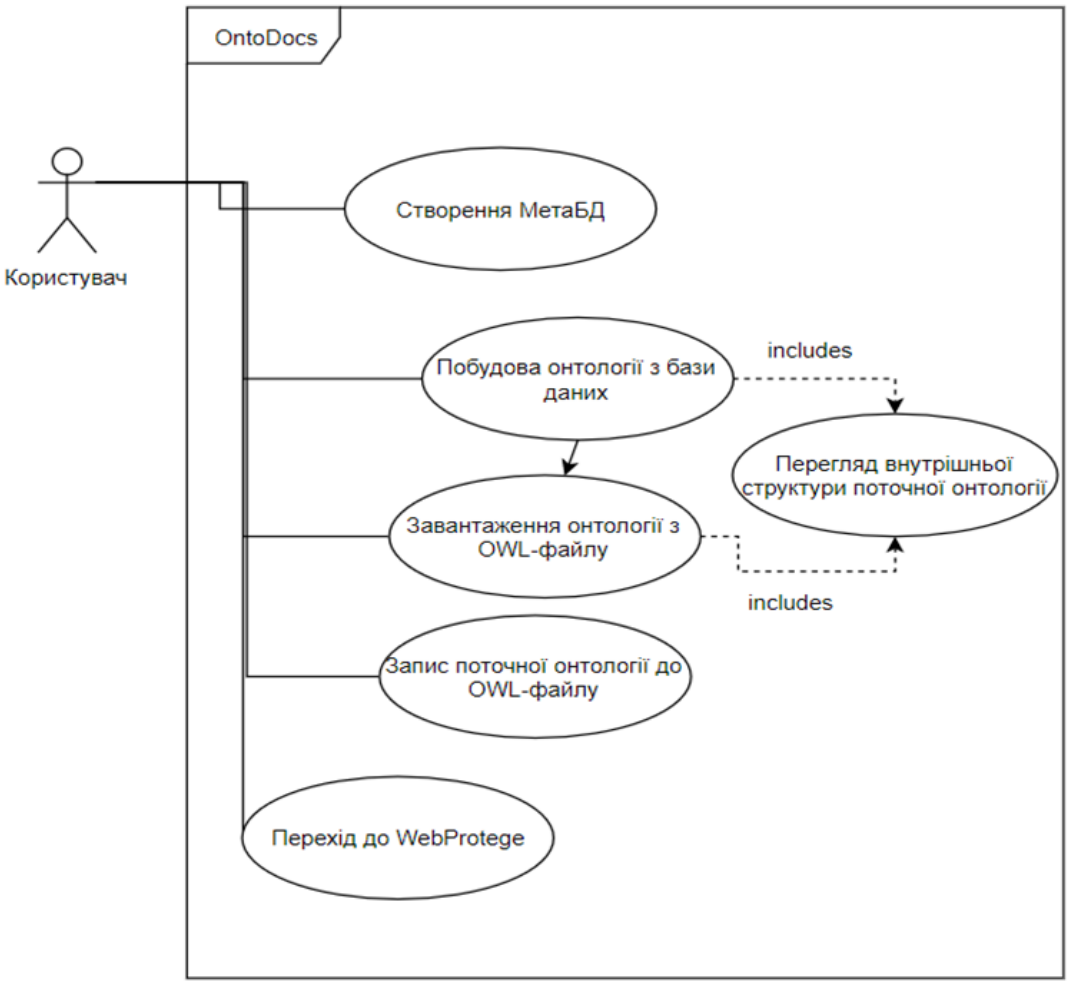

Рисунок 2 – Функціональна модель системи

У роботі з системою користувачу доступний такий функціонал:

1. Створення мета бази даних у середовищі MySQL.
2. Побудова структури онтології по заданому метаопису реляційного типу.
3. Завантаження онтології з OWL-файлу.
4. Перегляд внутрішньої структури поточної онтології.
5. Запис поточної онтології до OWL-файлу.

6. Перехід до WebProtege, де можна завантажити створені OWL-файли і продовжити роботу з онтологією використовуючи інструменти WebProtege.

1.3. Висновки до розділу

Для задачі формування реєстру необхідно мати наступну інформацію:

– які типи документів існують у предметній області;

– який розподіл по категоріям полегшить доступ до документів, що містять подібну інформацію;

– які особливості документів слід врахувати для віднесення їх до конкретних категорій.

2. АНАЛІЗ ПРОБЛЕМИ ОНТОЛОГІЧНОГО ПІДХОДУ ДО ПРОЕКТУВАННЯ РЕЄСТРУ ІНФОРМАЦІЙНИХ РЕСУРСІВ

Сьогодні при роботі з інформацією часто використовуються команди пошуку, форматування та збереження інформації для її подальшої обробки. Накопичені людством знання, як взагалі так і для конкретних галузей, складають основу для майбутніх досліджень.

Сформовані у вигляді документів ці дані можуть слугувати нормативно-правовою базою для виконання деяких дій, бути звітами щодо результатів їх проведення, взаємодіяти між собою, реалізовуючи механізм посилання та цитування.

Отже, інформаційні ресурси використовуються для різних цілей. Спосіб їх розміщення у сховищах даних досить часто суттєво залежить від направленості та галузі їх подальшого очікуваного використання.

Для невеликих проектів документна база може бути невпорядкованою або мати слабкий рівень організації. Якщо значні втрати на час пошуку все ж є допустимими, такий підхід має право на існування.

Серйозною проблемою задача доступу до таких ресурсів стає, коли їх кількість стрімко зростає або постає питання швидкого вибору тих з них, які потрібні для вирішення поточної задачі.

Впорядкування інформаційних ресурсів дозволяє вирішувати дану проблему. Ступінь успіху її вирішення напряму залежить від підходу, що використовується.

Онтологічний підхід використовується досить давно для широкого спектру завдань, що потребують впорядкування за певним критерієм.

Розуміння важливих особливостей даного підходу вкрай важливо для побудови програмної системи, яка його використовує.

2.1 Поняття інформаційних ресурсів та реєстру інформаційних ресурсів

Після кожної інформаційної революції виникало нове сприйняття інформації та інформаційних ресурсів. Вони набували іншу форму і значення для життя людства. Одночасно, кожна інформаційна революція збільшувала загальний обсяг інформаційних ресурсів і кількість людей, зайнятих їх обробкою.

Необхідно відзначити, що до середини минулого століття термін «інформаційні ресурси» практично не використовувався.

Протягом усієї історії людства впорядковані сховища знань, інформації називалися бібліотеками, архівами, банками даних і тощо. Тільки з появою комп'ютерної техніки виник термін «інформаційні ресурси», який згодом поширився і на традиційні форми зберігання і впорядкування інформації.

Отже, інформаційні ресурси — це джерела відповідним чином організованої інформації.

Зберігання та обробка інформації в комп'ютерних системах, обмін даними між ними та доступ до них користувачів неможливий без явно представлених описів властивостей цих даних. Ці описи необхідні програмним засобам, які виконують зазначені функції, а також користувачам для формулювання запитів, аналізу даних і інтерпретації їх змісту. Описи такого типу називаються метаданими, що і формує у нашому випадку реєстр інформаційних ресурсів [1].

2.2 Поняття онтології

Поняття онтології стосується усіх даних та відносин між ними, що існують для певного предмету досліджень.

У філософії так характеризують буття як таке. Очевидно, що його повний опис є неможливим, тому мова йде про сприйняття буття людиною-дослідником у розрізі його власного світосприйняття.

Онтологія містить усі найважливіші дані, з яких можна сформувати чітке та достатньо повне уявлення про предмет досліджень, яке необхідне для проведення цього дослідження.

Отже, зміст кожної онтології залежить від сутності дослідження, для проведення якого вона створюється.

Опис елементів та відношень онтології може виконуватися у простому описовому текстовому форматі, однак часто для цієї мети використовують графи.

Граф є дискретною структурою, що складається з вершин та ребер між ними.

Окремим видом графу є дерево. Ребра дерева мають стрілки на одному кінці (орієнтованість) та не утворюють циклів, що відображає рис.3.

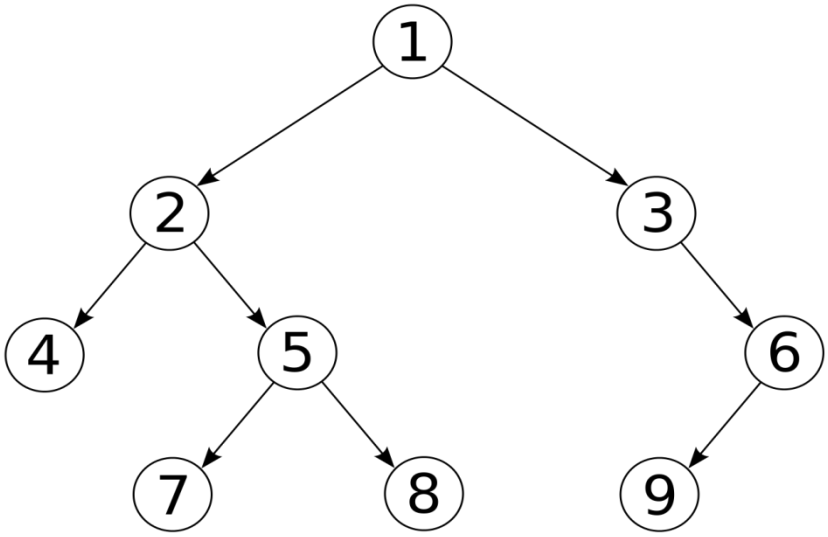

Рисунок 3 – Приклад графічної візуалізації структури у формі дерева

Дерево є вдалим вибором для відображення ієрархічних відношень між компонентами. Зауважимо, що зазвичай такий тип відношень є найбільш поширеним в онтологічних схемах.

З точки зору інформатики онтологія предметної області задається як множина всіх класів та їх екземплярів, що можуть бути виділені у складі цієї предметної області, та різноманітних (не лише ієрархічних) відношень між ними, обмежень на значення їх атрибутів.

Типовими складовими онтологічної схеми є:

— екземпляри — складові, що розглядаються дослідженням на найнижчому рівні;

— класи — сукупності екземплярів, що мають подібні спільні ознаки;

— атрибути — властивості екземплярів;

— відношення — закономірностей функціонального поєднання окремих екземплярів з різних класів або окремих класів на найвищому рівні.

Онтологія предметної області у такому форматі запису може бути формалізована та оброблена обчислюваною технікою.

Задача побудови онтології для предметної області є важливою складовою створення програмної системи для роботи з нею вже на ранніх етапах аналізу та планування.

Для стандартизації створення, обміну та реалізації онтологічної інформації було розроблено низку стандартів. Їх використання пришвидшує процес аналізу та дозволяє уникати проблем взаємного сприйняття у випадку використання двох та більше засобів, вхідною інформацією для яких слугує онтологічна схема предметної області.

До найбільш популярних стандартів мов онтологій належать:

— DOGMA,

— OIL,

— OWL.

Також існує стандарт для виконання запитів до онтологій.

Ці та інші стандарти рекомендовані консорціумом WWW для використання при побудові семантичних Інтернет-ресурсів.

2.3 Мова опису онтологій OWL

Мова веб-онтології (OWL) — це мова для визначення і створення веб-онтологій (Рекомендацій W3C) [3]. Онтологія OWL включає опис класів, властивостей і їх примірників. OWL використовується для явного подання сенсу термінів у словниках і зв′язків між цими термінами. Семантична мережа

(революційне бачення Тіма Бернерса-Лі) — це не що інше, як веб-сайт другого покоління. Він об'єднує мережу інформації, яка дозволяє підвищити ефективність, розширити обмін знаннями та простоту її використання. Онтології є ключем до цієї функціональної сумісності, оскільки вони визначають мову, з яким програмні агенти повинні спілкуватися один з одним, і людям потрібно спілкуватися з агентами. Семантична мережа дозволить автоматично збирати і зіставляти різні частини інформації про об'єкти, доступні в різних веб-ресурсах. Семантична мережа зберігає наш час, який ми витрачаємо на навігацію з одного веб-ресурсу на інший, щоб отримати інформацію про конкретний об'єкт.

Стандарт мови Web Ontology Language (OWL) використовується системою як базовий для опису онтологій у XML-документах.

Формат XML (eXtensible Markup Language) дозволяє створювати користувачеві схеми документів для своїх потреб. У основі схеми лежать елементи, які можуть містить в собі інші елементи або прості дані. Існує багатий арсенал обмежень, які можна застосовувати до вмісту елементів та значень простих даних.

Назва онтології у мові OWL позначається URL, що полегшує її ідентифікацію.

Можна виділити 3 підмножини мови OWL, що відрізняються рівнем абстракції опису онтологій:

— OWL Lite,

— OWL DL,

— OWL Full.

Засоби OWL Lite дають лише поверхневе уявлення про предметну область.

Підмножина OWL DL є достатньою для виконання переважної більшості задач, які використовують онтології.

Набір OWL Full забезпечує надзвичайно високу виразність, проте це вимагає суттєвого зменшення швидкодії при обробці написаних з його використанням файлів.

Формат RDF (Resource Description Framework), який було створено для опису метаданих у Інтернеті, слугує основою для мови OWL.

Існує також низка стандартів, які використовують OWL- файли для побудови вищого рівня онтологічної абстракції або виконання запитів.

Мова запитів SPARQL є найвідомішим стандартом. Вона дозволяє створювати запити до онтологій декількох типів та оброблювати результати у вигляді триплетів, де екземпляри онтологій поєднані предикатом відношення.

2.4 Аналіз існуючих програмних систем

Проблема взаємодії редакторів онтологій та реляційних баз даних, а саме формату мови OWL та реляційної моделі все ще постає досить гостро, оскільки існує дуже мало рішень, які б могли задовольнити її вирішення для кожного випадку.

Серед існуючих підходів конвертації між даними owl-формату та інформацією, збереженою у реляційних базах даних, визначимо такі:

— до існуючої онтології заносяться дані з бази даних;

— на базі структури бази даних будується структура онтології, вузли якої відповідають таблицям БД.

2.4.1. Платформа D2RQ

Плаформа D2RQ — інтегроване середовище для доступу до інформації, що зберігається у реляційних базах даних, у вигляді віртуальних RDF-графів, які доступні лише для перегляду. Серед можливостей середовища:

1. Опис зв’язку між онтологією та реляційною базою даних.

2. Використання SPARQL-запитів до бази даних.

Однак, вагомим недоліком середовища, є складний для сприйняття користувачів інтерфейс.

На рисунку 4 зображено схему роботи середовища D2RQ.

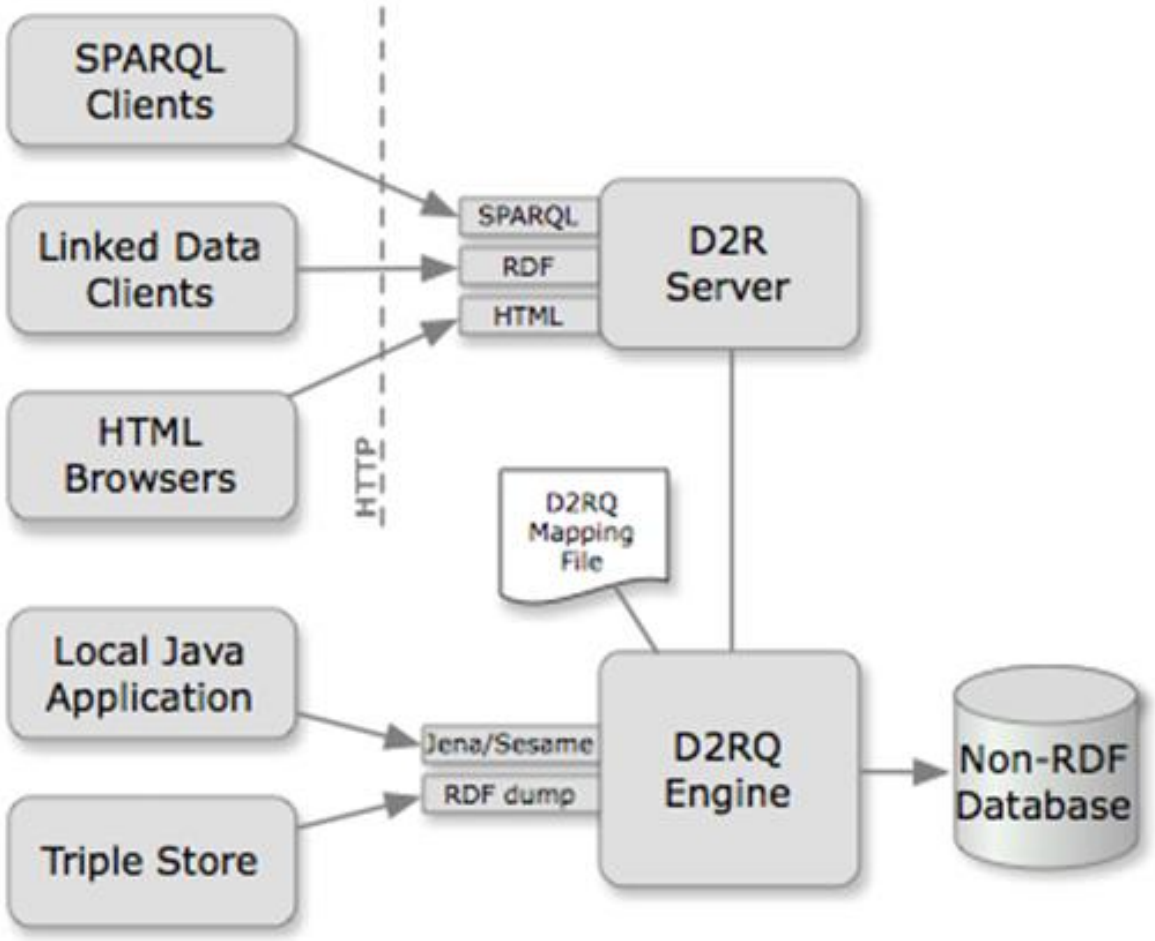

Рисунок 4 – Схема роботи середовища D2RQ

2.4.2. Маппінг – мова R2RML

Маппінг – мова R2RML, що визначає, яким чином відображати інформацію, що зберігається у реляційних базах даних, у вигляді RDF триплетів.

Встановлення таких зв’язків, надає можливість переглядати дані реляційних баз даних у моделі даних RDF з такою структурою, яку обирає користувач.

Спосіб встановлення зв’язків у R2RML часто є складним для розуміння користувача, що може вплинути на хід роботи та викликати помилки у представленні інформації як RDF триплетів.

На рисунку 5 зображено сценарій роботи R2RML.

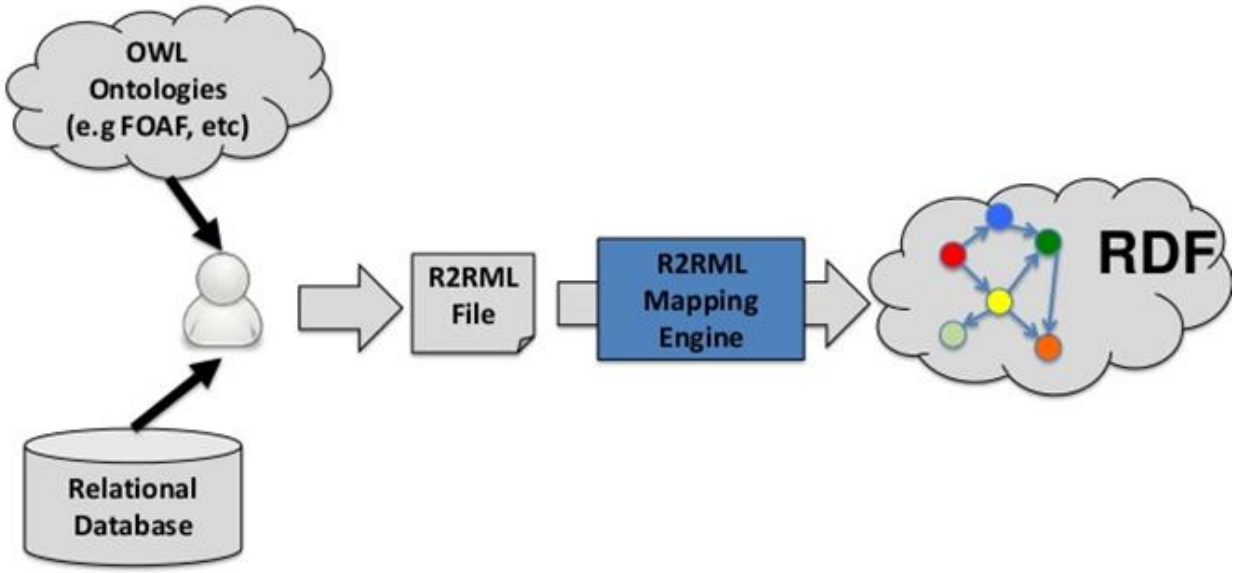

Рисунок 5 – Сценарій роботи R2RML

2.4.3. Алгоритм OWL2DB

Процес перетворення OWL-файлу до формату реляційної бази даних був формалізований групою дослідників Масачусетського університету США під керівництвом Анурадхи Галі [14].

Алгоритм було побудовано на основі широкого спектру методик перетворення OWL-файлів до чистого XML-формату.

Було показано ефективність отриманих ними попередніх результатів, однак не було створено кінцевого продукту, який би реалізовував їх відкриття.

Однією з причин цього можна назвати достатню складність запропонованого ними підходу для реалізації існуючими програмними засобами.

2.5. Висновки до розділу

У даному розділі було описано сутність понять «інформаційні ресурси», «реєстр інформаційних ресурсів».

Дано визначення поняття «онтологія», розглянуто існуючі формати опису онтологій та аналогію

Проаналізовано існуючі методи представлення інформації, збереженої у реляційній базі даних у вигляді онтології.

Серед існуючих підходів конвертації між даними owl-формату та інформацією, збереженою у реляційних базах даних, переважають 2 підходи:

— до існуючої онтології заносяться дані з бази даних;

— на базі структури бази даних будується структура онтології, вузли якої відповідають таблицям БД.

Систем, які за заданим метаописом об’єктів предметної області створюють структуру онтології не виявлено.

3. ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

Для побудови ефективної системи необхідно залучати достатній обсяг новітнього програмного забезпечення та вже реалізованих рішень, які можна інтегрувати до цієї системи з метою підвищення її швидкодії та зручності використання.

Наприклад, різні мови програмування мають неоднакову потужність у контексті вирішення конкретної задачі програмування, оскільки мають власні системи бібліотек та шаблонів реалізації.

З іншого боку, сучасні мови програмування підтримують велику кількість різних парадигм програмування, що дозволяє використовувати одну з них для вирішення усього комплексу задач, не витрачаючи додатковий час на інтеграцію модулів системи, що було створені з використанням принципово різних стеків технологій.

Налаштування проекту та використання зовнішніх пакетів, створених іншими користувачами, стає легшим за умови використання спеціалізованих середовищ розробки та засобів управління пакетами.

До створення системи, що пропонується, було залучено наступні програмні засоби розробки:

— систему управління реляційними базами даних MySQL;

— середовище для зручної роботи з базами даних MySQL Workbench;

— мова програмування C#;

— бібліотека для створення графічного інтерфейсу користувача WindowsForms;

— фреймворк ADO .NET для взаємодії з реляційними базами даних;

— фреймворк DotNetOwlAPI для роботи з онтологією у вигляді графу;

— засіб управління пакетами NuGet;

— середовище розробки Visual Studio.

3.1. Система керування реляційними базами даних MySQL

Бази даних використовуються для тимчасового або довгострокового збереження даних у системі. Існують різні формати баз даних, наприклад:

— ієрархічні (дані у вигляді вершин дерева);

— графічні (дані у вигляді вершин довільного графу);

— реляційні (дані у таблицях);

— документні (дані у JSON-документах).

Для кожного формату існують відповідні системи керування базами даних.

Системи керування реляційними базами даних відрізняються найбільшою надійністю, оскільки підтримують режим транзакцій. Під транзакцією розуміємо послідовність дій, які або виконуються усі, або не виконується жодна з них. Використання транзакцій при виконанні операцій зміни даних у базі означає протидію можливій шкідливій ситуації часткової зміни, коли апаратна або програмна помилка сталася безпосередньо при виконанні цілісного набору запитів до зміни.

Більшість даних систем використовує для керування базами даних мову запитів SQL (Structured Query Language). Можливості мови SQL дозволяють виконувати велику множину дій, у тому числі:

— створення нової бази даних;

— створення таблиці у існуючій базі даних;

— заповнення таблиці новими даними у вигляді рядків;

— редагування існуючих рядків;

— безпечне видалення рядків;

— виконання запитів до однієї або декількох таблиць, використовуючи складні критерії умови та деякі елементарні шаблони.

Реляційна модель даних складається з табличок та обмежень на дані. Таблиця відображає структуру сутності предметної області, кожен її стовпчик може містити дані тільки одного типу.

Кожна таблиця повинна мати первинний ключ, тобто стовпець, на основі змісту якого можна буде ідентифікувати рядки. Дані у цьому стовпці повинні бути унікальними, тобто не можуть повторюватися.

Таблиці сполучені між собою за допомогою зовнішніх ключів, тобто стовпчиків однієї таблиці, які містять посилання на стовпчики іншої (зазвичай, на первинний ключ).

Система керування базами даних MySQL, яку було обрано для збереження онтології у реляційному вигляді, є надзвичайно поширеним рішенням для невеликих проектів та має наступні переваги:

— оптимізація операцій виконання певних типів запитів та доступу до індексів;

— наявність документації з детальним описом можливостей системи;

— легке налаштування для різних операціних систем та апаратних платформ.

Система автоматично створює користувача «root», але є можливість створення власних користувачів та налаштування їх прав для доступу та редагування баз даних. Засіб MySQL Workbench (рис. 6) надає можливості графічного інтерфейсу для виконання взаємодії зі встановленим локальним сервером MySQL та підтримує плагіни (розширення) для аналізу ефективності запитів, побудови діаграм тощо.

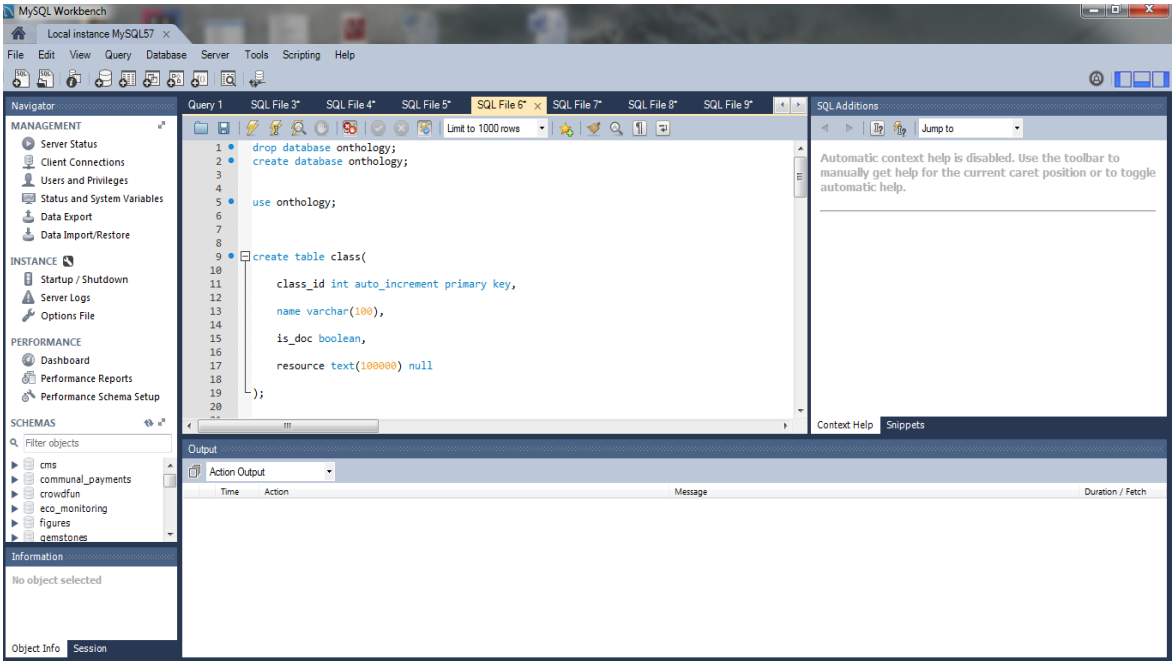

Рисунок 6 — Інтерфейс середовища MySQL Workbench

3.2. Мова програмування C#

Мова програмування C# була обрана для реалізації переважної частини програмних модулів системи, оскільки має ряд переваг над аналогами, зокрема підтримує більшість парадигм та стилів програмування, для неї розроблено широкий спектр бібліотек та фреймворків, існує якісна та детальна документація для роботи з нею.

Дана мова є популярним засобом для створення проектів різного масштабу та цільового призначення, не залежить від специфіки предметної області та подібних деталей, надає гнучкі та потужні вбудовані рішення типових проблем, що виникають під час програмування.

Функціональною одиницею програмного коду на цій мові є клас, який може містити поля даних (зазвичай у форматі властивостей, що реалізують механізм укриття) та методи.

Класи знаходяться у просторах імен, де їх імена мають бути унікальними для однозначної ідентифікації.

Мова C# була створена компанією Microsoft на початку нового тисячоліття та продовжує розвиватися відповідно до вимог науково-технічного прогресу у програмуванні. Зокрема, активно створюються засоби для роботи зі хмарними сховищами даних та розподіленими у них обчисленнями.

3.3. Бібліотека Windows Forms

Windows Forms — назва інтерфейсу програмування додатків (API), що відповідає за графічний інтерфейс користувача і є частиною Microsoft .NET Framework.

Windows Forms спрощує доступ до візуальних компонентів (віджетів) Microsoft Windows за рахунок створення обгортки для існуючого Win32 API в керованому коді.

Керований код є машинно-незалежним і не залежить від мови розробки. Тобто програміст однаково може використовувати Windows Forms як при написанні програмного додатку на C #, C ++, так і на VB.Net, J # і ін.

Основний клас Windows Forms — форма, екземплярами якого є головні і діалогові вікна. Форми є нащадками класу Form, визначеного в просторі імен System.Windows.Forms. Інтерфейс Windows Forms дозволяє працювати в режимі конструктора, додаючи елементи управління (кнопки, поля для введення тексту, поля для відображення тексту, меню та інші компоненти, характерні для Windows—додатків) простим «перетягуванням». У коді вони представлені як поля класу форми. Всі елементи управління вікна є об'єктами класів, що містяться в System.Windows.Forms і є нащадками базового класу Control. Відповідна реакція програми на певну дію (подія), пов'язане з елементом управління — обробник події — оформляється у вигляді методу форми.

Для системи, що пропонується, було використано засоби Windows Forms, оскільки вони простіші у використанні та не потребують вивчення спеціалізованої розмітки.

Модель, що використовується у Windows Forms, складається з графічних елементів (рисунок 7), які можна розміщувати у робочій області (формі), задаючи їх атрибути (шрифт тексту, колір тощо), та подій, які виникають при взаємодії користувача з певними елементами під час роботи програми.

Кожна подія обробляється власним методом, який містить довільний код на мові програмування C# та може впливати на стан або атрибути інших графічних елементів.

Простота та потужність даного рішення дозволяють швидко зробити дизайнерський макет застосунку та надати йому програмні засоби імплементації у разі узгодження.

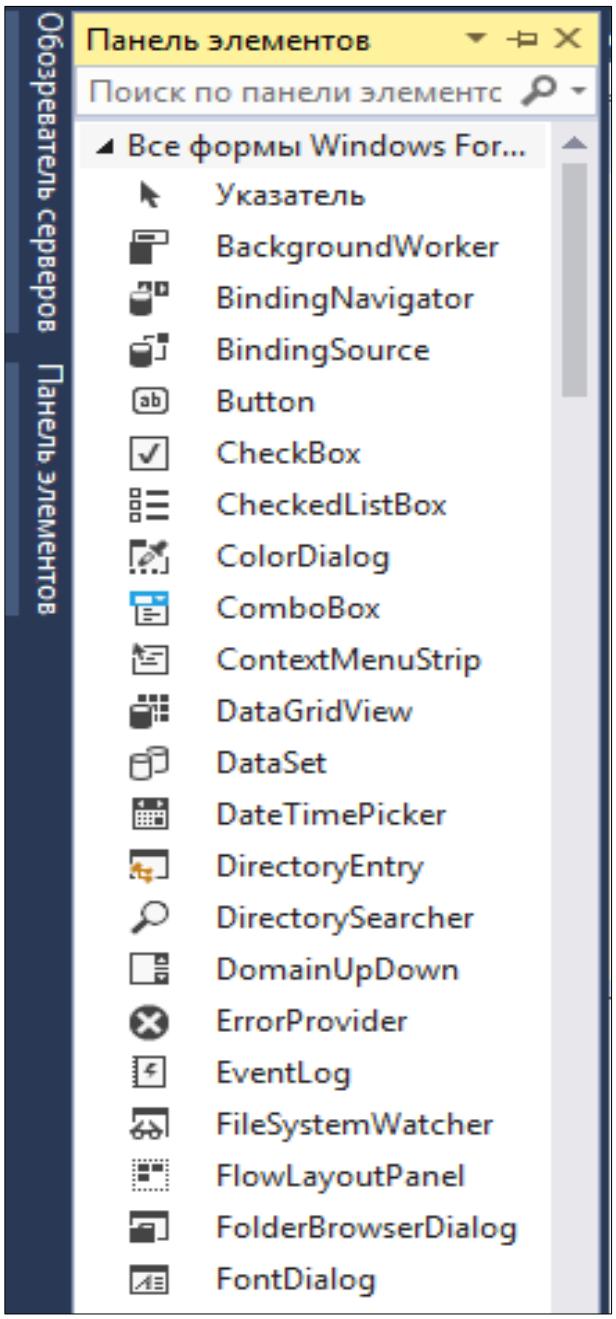

Рисунок 7 — Панель графічних елементів бібліотеки Windows Forms

3.4. Фреймворк ADO .NET

Для взаємодії системи, написаної засобами мови C#, з реляційними базами даних існує універсальне рішення ActiveX Data Object (ADO) для .NET.

Платформа .NET використовується для декількох мов програмування та компілює їх у середовищі Common Language Runtime, що дозволяє поширити один підхід одразу на всі мови.

Стандартна версія ADO може використовуватися для роботи з різними джерелами даних, наприклад:

— Microsoft SQL Server;

— Microsoft Access;

— Microsoft Excel;

— Microsoft Outlook;

— Microsoft Exchange;

— Oracle;

— протокол ODBC;

— формат даних XML.

У фреймворку реалізовано універсальний підхід до роботи з джерелом даних.

Процес виконання запитів та програмної обробки результатів складається з декількох послідовних етапів.

На початку створюється, налаштовується та відкривається екземпляр підключення, ініціалізація якого відбувається текстовим рядком, де перелічені всі необхідні параметри та їх значення.

Потім створюється та налаштовується екземпляр команди для відкритого підключення, який містить інформацію про запит.

Викликається метод ExecuteReader () для команди, щоб вона була відправлена на виконання та синхронно повернула результати у вигляді екземпляру зчитувача.

Дані повертаються рядками з екземпляру зчитувача, при цьому вже можливе їх довільне форматування та подальше використання програмою.

Використання ADO .NET для взаємодії з MySQL можливе за умови завантаження розширення MySql.Data через менеджер пакетів NuGet (рисунок 8) та налаштування драйверу цієї системи управління реляційними базами даних.

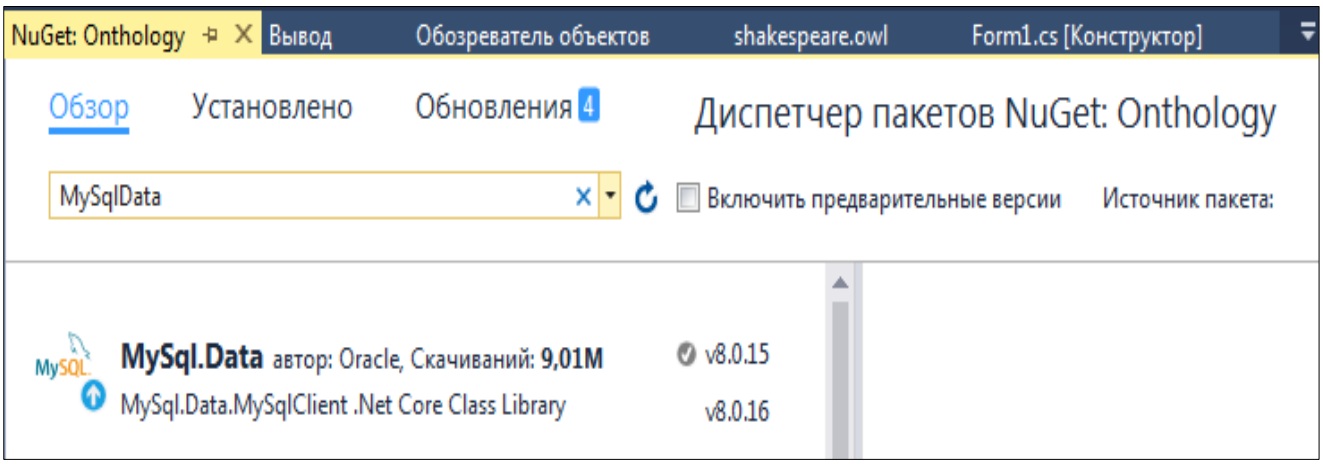

Рисунок 8 — Розширення MySql.Data у вікні диспетчеру пакетів NuGet

3.5. Фреймворк DotNetOwlAPI

Для взаємодії з файлами онтологій та проміжного представлення їх у системі у вигляді графу було використано рішення DotNetOwlAPI.

Даний фреймворк містить засоби для:

— експорту та імпорту онтологічних графів до XML-документів;

— обходу вершин онтологічних графів;

— виконання SPARQL-запитів до графів;

— безпосереднього редагування структури та вмісту онтологічних графів.

Для використання цієї бібліотеки у проекті було завантажено фреймворк більш широкого застосування dotNetRDF (рисунок 9), який, у свою чергу, містить DotNetOwlAPI як важливу складову частину.

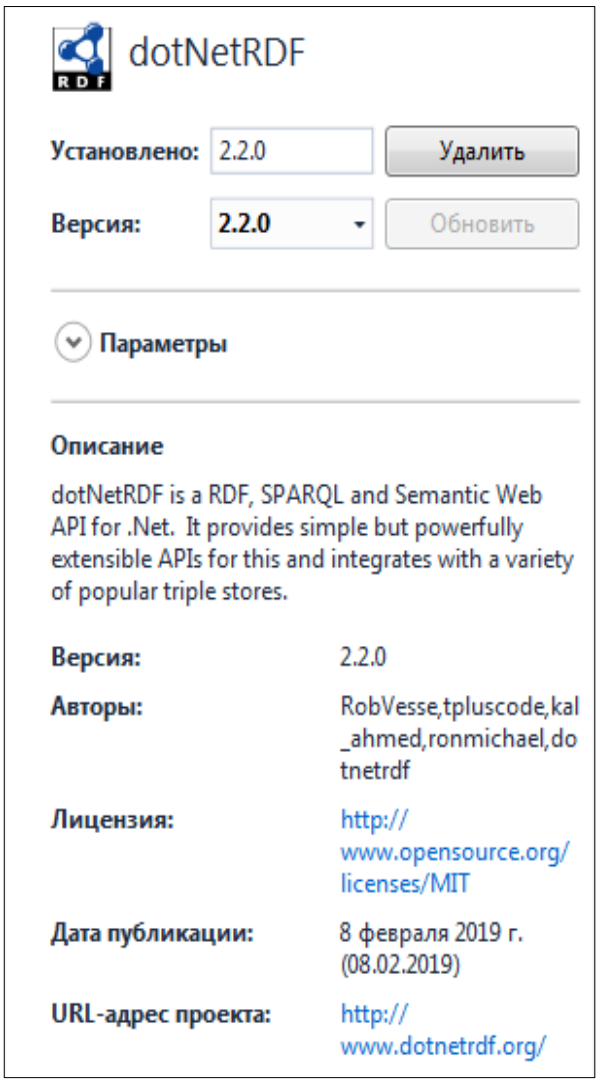

Рисунок 9 — Розширення dotNetRDF у вікні диспетчеру пакетів NuGet

3.6. Середовище розробки Visual Studio

Для написання програмного коду системи на мові C# було обрано середовище Visual Studio.

Середовище розробки Visual Studio (рисунок 10) може бути використане для створення різноманітних проектів на мовах платформи .NET.

Перевагами середовища над аналогами є:

— підсвітка синтаксису та авто доповнення;

— засоби аналізу ефективності виконання програмного коду;

— власний відлагоджувач;

— якісна документація.

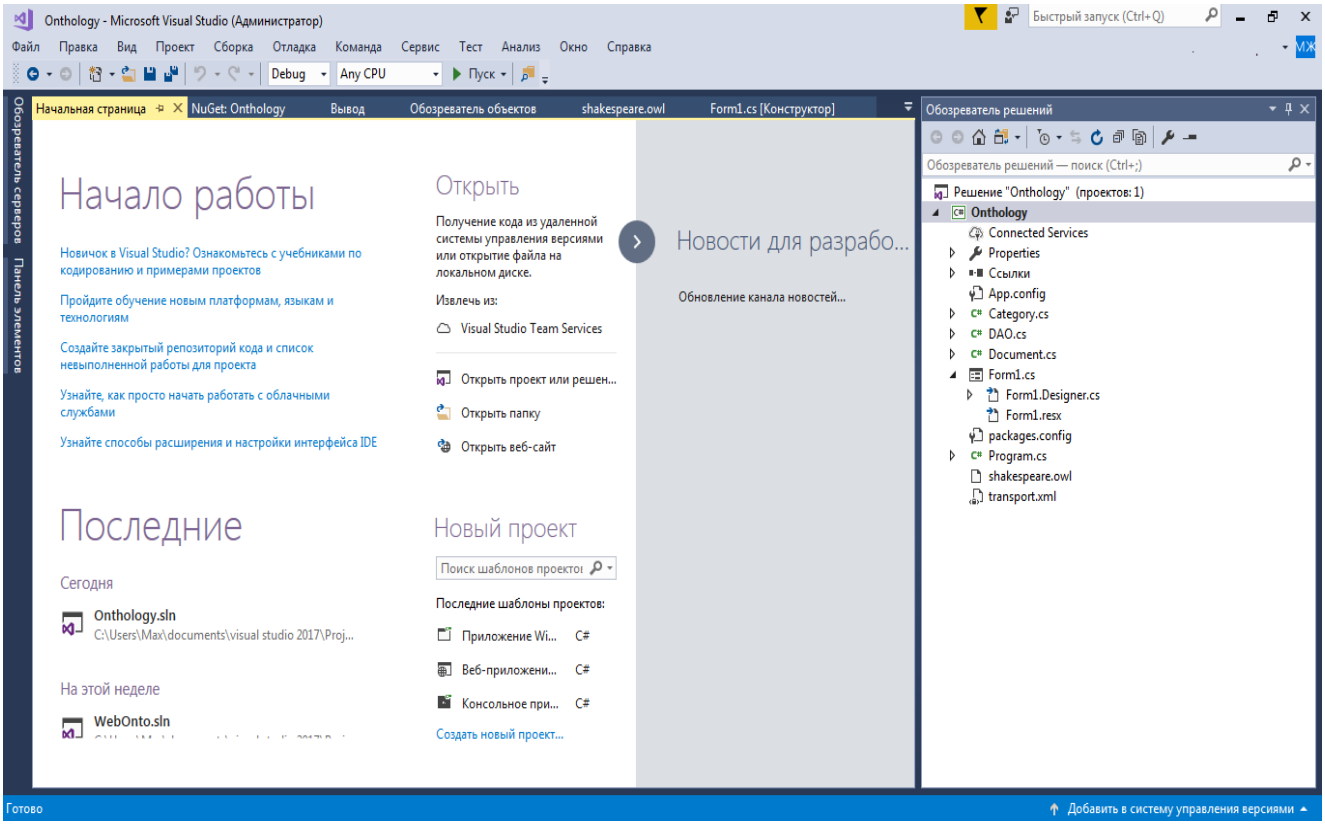

Рисунок 10 — Інтерфейс Visual Studio 2017

3.7. Висновки до розділу

В даному розділі були описані основні програмні засоби розробки, використані для побудови системи, що пропонується, та їх переваги над існуючими аналогами.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Розроблена система містить два складні інтерфейси для взаємодії із зовнішніми даними різних форматів:

— реляційною базою даних MySQL;

— онтологією формату мови OWL у документі XML.

Загальний вигляд системи зображено на рисунку 11.

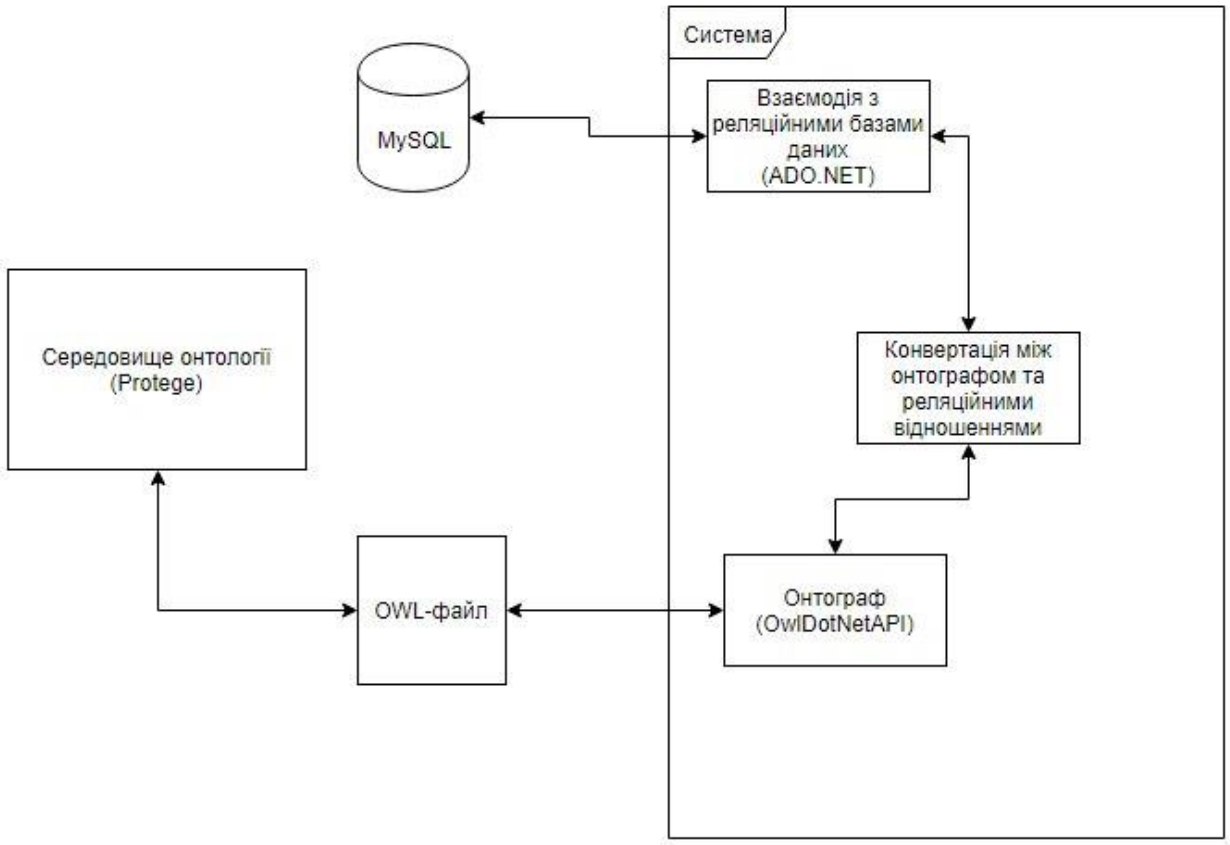

Рисунок 11 — Принципова архітектура розробленої програмної системи

Як видно з даного рисунку, архітектура системи повністю відповідає функціоналу, який вона покликана вирішувати.

Для потреб взаємодії із зовнішніми джерелами збереження даних було використано готові рішення у вигляді фреймворків ADO .NET та dotNetRDF (DotNetOwlAPI), в той час як основна бізнес-логіка системи, яка відповідає за конвертацію між двома форматами даних та взаємодію з користувачем, була

створена із використанням бібліотеки для створення зручного графічного інтерфейсу користувача Windows Forms.

4.1. Структура бази даних

Для збереження онтології у базі даних було обрано схему з двох таблиць, тобто окремо зберігаються сутності та метадані про відносини ієрархії між ними (рисунок 12).

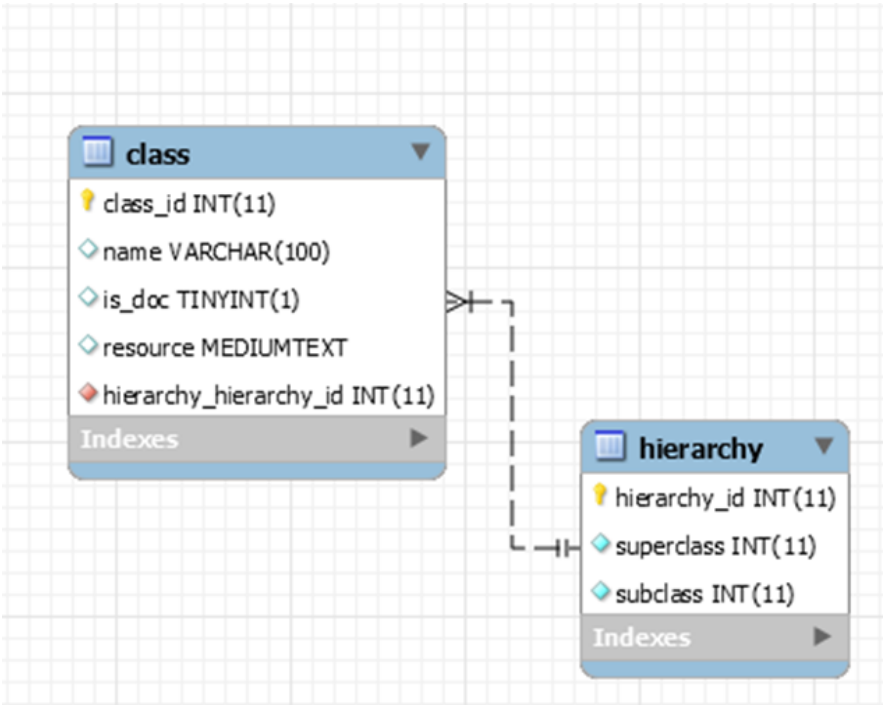

Рисунок 12 — Таблиці бази даних системи

Фактично було реалізовано концептуальне рефлексивне відношення «багатьох до багатьох» сутності «Клас» самої до себе. Для реалізації такого типу відношень створюється додаткова службова таблиця «ієрархія». У таблиці сутності класу зберігаються назви категорій та документів, відповідні логічні позначення, чи є конкретний об’єкт документом, та, у разі істини цього твердження, його текст.

Таблиця сутності ієрархії містить два зовнішніх ключа до першої таблиці та представляє відношення ієрархії між категоріями та входження документу до

конкретної категорії у формі пар ідентифікаторів конкретних об’єктів предметної області.

Використання лише двох таблиць для збереження даних про онтології будь-яких предметних областей, що не залежить від кількості сутностей та деревовидності структури онтології, є надзвичайно потужним рішенням проблеми ефективних запитів до великих онтологічних схем.

4.2. Структура програмного застосунку

Програмна система складається з кількох основних класів, відношення використання між якими схематично показано на рисунку 13.

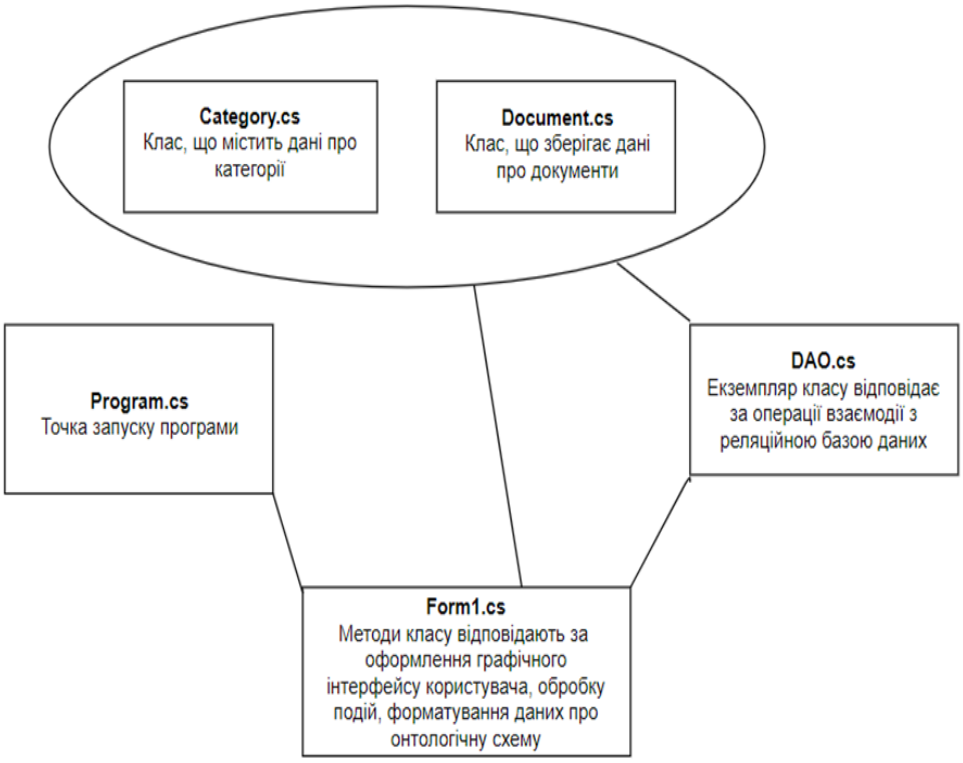

Рисунок 13 — Класи програмного застосунку та взаємозв’язки між ними

4.2.1. Модуль взаємодії з реляційною базою даних

Метадані, що описують класифікацію та структуру інформаційних ресурсів зберігаються в реляційній базі даних. Такий спосіб зберігання є зручним та компактним, але не дає користувачу широкого огляду організації взаємозв’язків між

класами інформаційних ресурсів. А при зростанні кількості об’єктів в метабазі даних взагалі уявити структуру та їх ієрархією.

Даний модуль взаємодії реляційної бази даних з онтологією дозволяє побудувати графічну структуру онтології по заданому опису об’єктів, що зберігається у базі даних.

Існуючі підходи до побудови структур онтологій з структури реляційної бази ґрунтуються на тому, що вузли (класи) онтології та їх атрибути повністю або частково відповідають таблицям реляційної бази даних.

У нашому ж випадку опис класів зберігається в двох таблицях. І на базі даних реляційної БД може бути побудована досить розгалужена та об’ємна структура онтології.

Клас DAO (Data Access Object) відповідає за операції взаємодії системи та її поточної онтології з базою даних системи MySQL.

Клас DAO реалізований за допомогою фреймворку ADO .NET та розширення MySql.Data. Клас містить низку методів:

— private void CreateConnection() для створення підключення до бази даних;

— public void TestConnection() для тестування встановленого підключення;

— public List<Category> LoadFromDB () для безпосереднього завантаження онтології з бази (із заміщенням поточної онтології);

— public void SaveToDB(List<Category> categories) для збереження поточної онтології до бази (із заміщенням онтологічної схеми, яка на той момент існує у базі даних).

Екземпляр класу DAO створюється та обслуговується у модулі інтерфейсу кінцевого користувача.

4.2.2. Класи сутностей «Категорія» та «Документ»

Сутності поточної онтологічної схеми зберігаються у системі за допомогою двох класів сутностей.

Клас Document, що зберігає дані про документи, містить конструктор за замовчуванням та перелік полів:

— public string ID { get; set; }, ідентифікатор;

— public string Name { get; set; }, назва;

— public string Content { get; set; }, текст;

— public Category Category { get; set; }, категорія документу.

Клас Category, що містить дані про категорії, містить конструктор за замовчуванням, в якому відбувається створення списків, та перелік полів:

— public string ID { get; set; }, ідентифікатор;

— public string Name { get; set; }, назва;

— public List<Category> Supercategories { get; set; }, перелік надкатегорій для цієї категорії;

— public List<Category> Subcategories { get; set; }, перелік підкатегорій для цієї категорії;

— public List<Document> Documents { get; set; }, перелік документів для цієї категорії.

4.2.3. Модуль взаємодії з кінцевим користувачем та роботи з OWL-файлами

Даний модуль призначений для побудови графічного відображення структури онтології.

При завантаженні OWL файлу до системи створюється екземпляр класу Graph, дані для якого зчитуються екземпляром службового класу OpenFileDialog та форматуються відповідним чином.

Клас Graph має метод LoadFromFile(filePath: String), для парсингу RDF-структур, у тому числі OWL-файлів. Дані відповідно зберігаються у форматів класів сутностей системи.

Одразу створюється графічне представлення онтології. Для цього використовується екземпляр класу TreeView бібліотеки WindowsForms.

Клас OpenFileDialog також надає можливості для запису поточної онтології до файлу, обраного кінцевим користувачем системи. Процес запису є обернено

аналогічним до процесу зчитування, оскільки клас Graph містить метод, що автоматизує цей процес.

Методи класу Form1, що були створені, відповідають за оформлення графічного інтерфейсу користувача, обробку подій, форматування даних про онтологічну схему, з якою в конкретний час працює програмна система. Екземпляр класу DAO відповідає за операції взаємодії з реляційною базою даних. Методи даного класу викликаються при обробці специфічних подій графічного інтерфейсу.

Отже, клас-форма Form1 є головною точкою інтеграції та взаємодії між основними компонентами створеної системи.

4.2.4. Алгоритм відображення онтології з реляційної моделі до OWL-файлу

На рисунку 14 приведена схема алгоритму відображення онтології з реляційної моделі до OWL-файлу.

Алгоритм відображення онтології з реляційної моделі до OWL-файлу складається з таких основних кроків:

1. Встановлення підключення до бази даних.
2. Зчитування інформації про категорії, документи, ті ієрархічні відношення між ними засобами ADO.NET.
3. Збереження отриманих даних у вигляді колекцій класів Category та Document.
4. Конвертація колекцій у граф засобами dotNetRDF.
5. Збереження графу у OWL-файл.

До таблиць реляційної бази даних складаються SQL-запити, отримана інформація зберігається у внутрішньому представлені системи.

До створеної порожньої структури онтологічного графу вносяться нові вузли категорій та документів. Потім, інформація з графу записується до owl-файлу.

У випадку конвертації онтології у реляційний формат, спочатку зчитається інформація з конкретного owl-файлу, яка зберігається у форматі внутрішнього представлення онтологічного графу.

Далі виконується крок проходження вершинами графу з виділенням класів та екземплярів. Виділенні дані зберігаються у внутрішньому представленні системи.

У базі даних ці дані зберігаються за допомогою SQL-інструкцій.

Рисунок 14. — Схема алгоритму відображення онтології з реляційної моделі до OWL-файлу

4.3. Висновки до розділу

В даному розділі були описані технічні деталі реалізації та архітектурні рішення, що були використані при побудові системи.

Дано опис моделі реляційної метабази даних, яка відповідає реєстру інформаційних ресурсів.

Дано опис системи побудови структури інформаційного ресурсу у вигладі онтології.

Система має можливість доповнювати метабазу даних новими даними, організованими у вигляді об’єктів онтології інформаційного ресурсу.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

При запуску програми OnthoDocs користувач на екрані бачить головну форму додатку (рисунок 15)

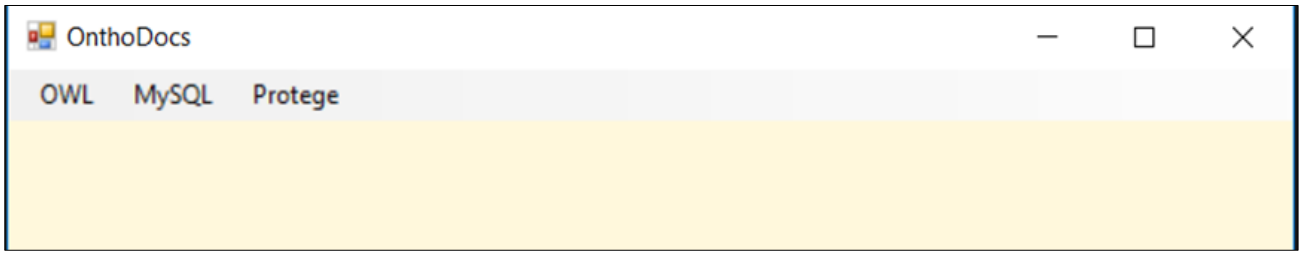

Рисунок 15 — Головна форма додатку

На головній формі розташовані основні три кнопки меню для роботи з додатком, а саме: «OWL», «MySQL», «Protege».

Натиснувши на кнопку MySql користувач має такі доступні дії:

— перевірити чи наявне підключення до бази даних;

— завантажити онтологію з MySql;

— зберегти онтологію у MySql.

Після натискання на кнопку “завантажити онтологію з MySql” у вікні користувача з’явиться збережена заздалегідь у базі даних онтологія. На рисунку 16 зображено меню роботи з MySql.

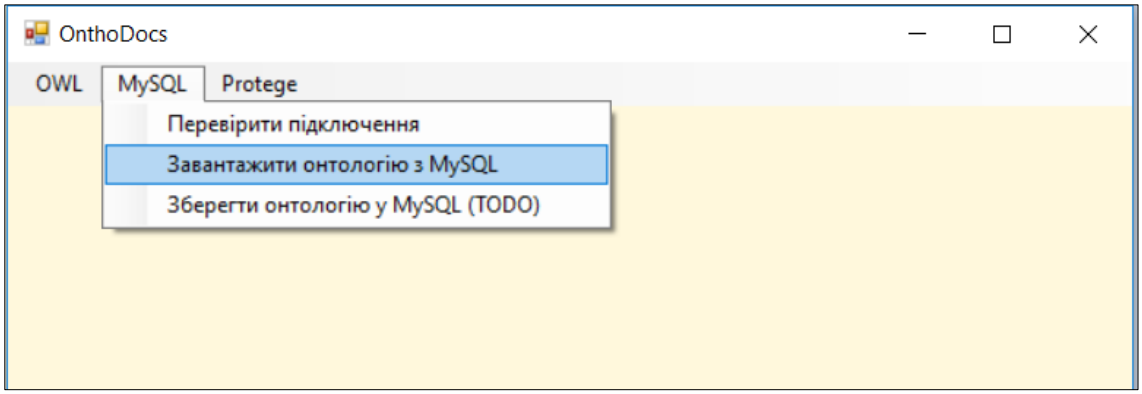

Рисунок 16 — Меню роботи з MySql

Збережені інформаційні ресурси, які знаходяться у базі даних, зображуються у вигляді ієрархії, а саме класи та підкласи онтології, що зображено на рисунку 17.

Рисунок 17 — Зображення онтології, завантаженої з бази даних

Також система має таку опцію, як збереження онтології у owl-файлі. Зображення на рисунку 18.

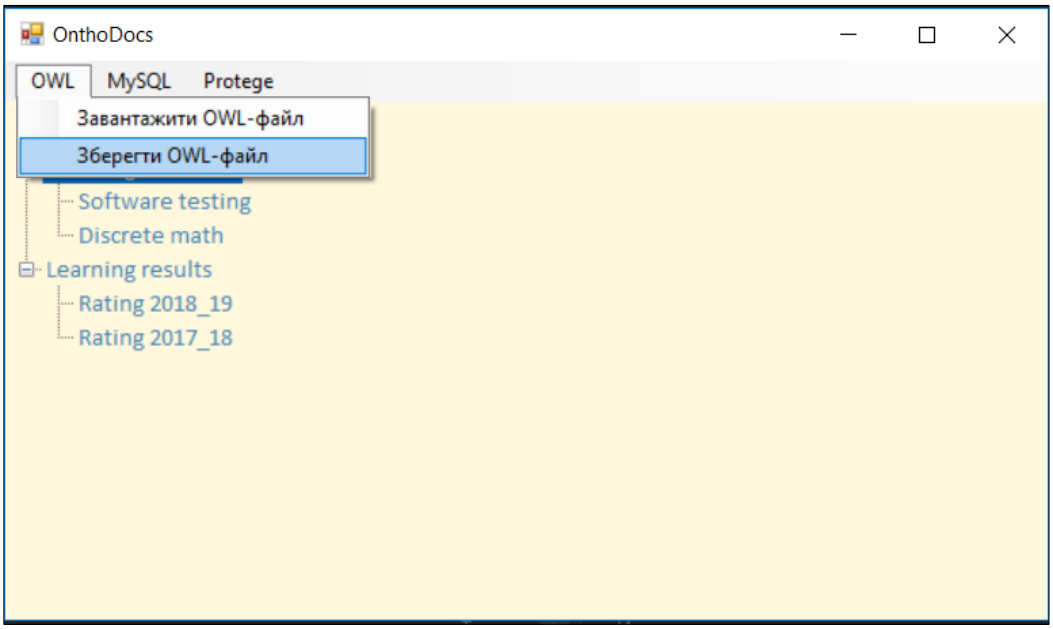

Рисунок 18 — Меню роботи з owl-файлами

Коли користувач натисне на кнопку “зберегти owl-файл”, він побачить вікно збереження файлу на свій комп’ютер, де зможе обрати місце збереження файлу, це демонструє рисунок 19.

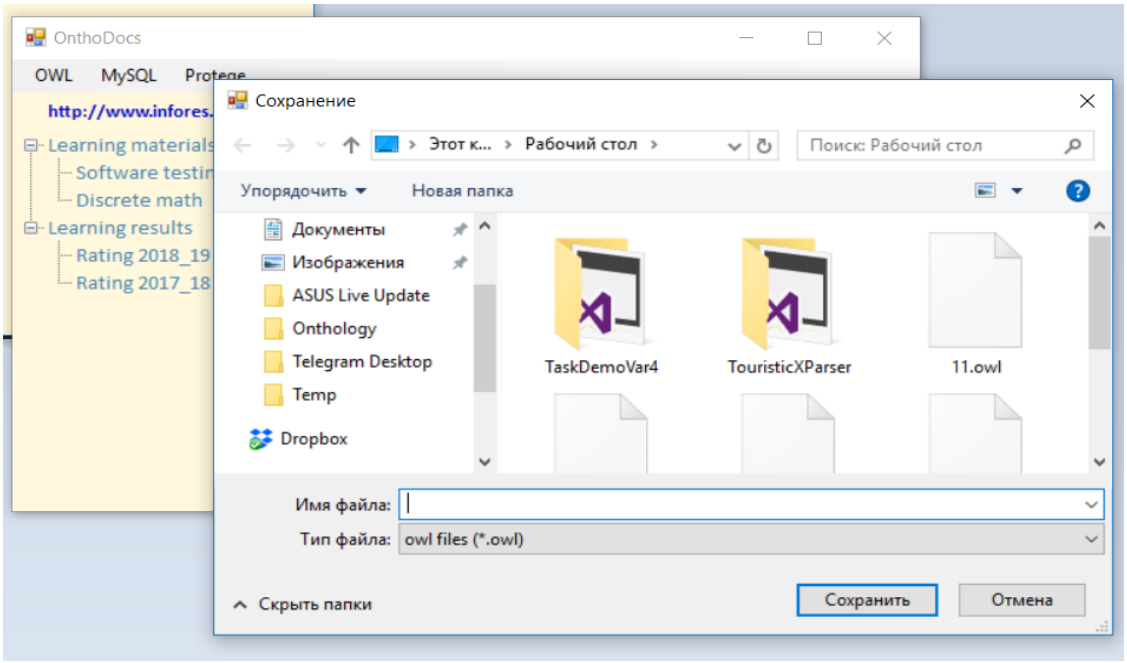

Рисунок 19 — Збереження owl-файлу на комп'ютер

Додаток також надає можливість користувачу працювати уже зі створеними owl-файлами. Для цього, у меню «OWL» потрібно обрати опцію “завантажити owl-файл”, що відображає рисунок 20.

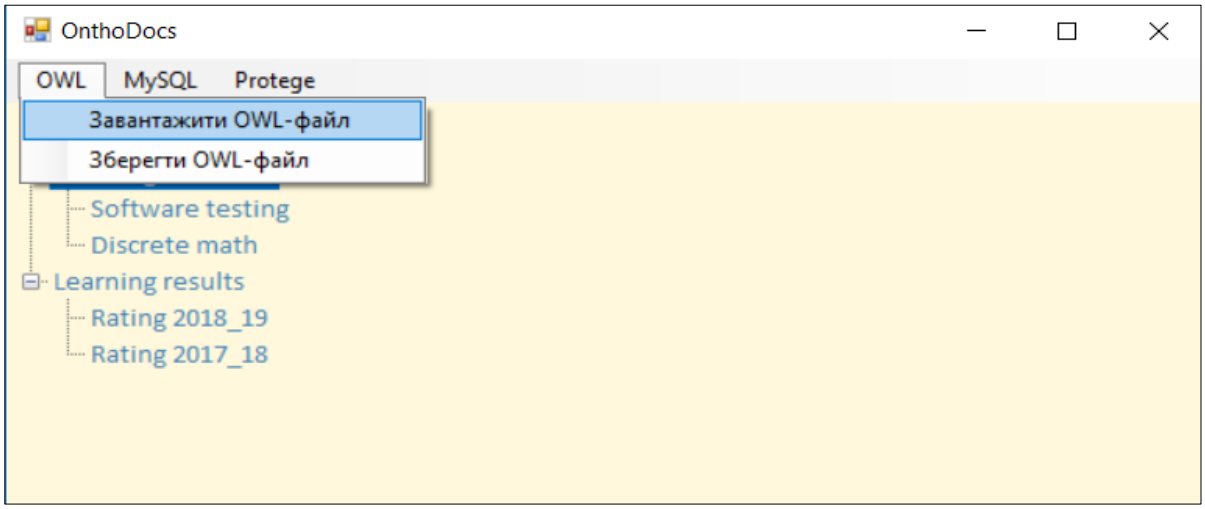

Рисунок 20 — Завантаження owl-файлу

Програма дозволяє обрати будь-який owl-файл, який знаходиться на комп'ютері користувача у вікні “вікриття”. Зображено на рисунку 21

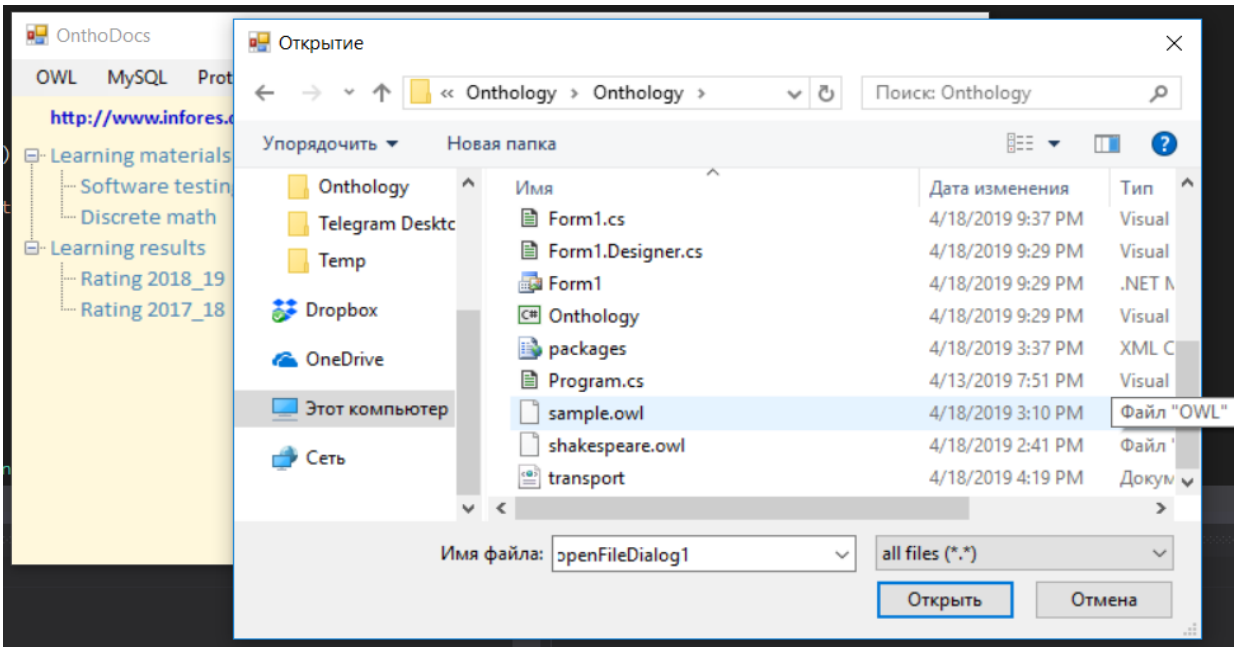

Рисунок 21 — Вибір owl-файлу з комп'ютера користувача

Коли користувач обрав owl-файл, який він бажає відкрити у додаткові, програма відображає дану онтологію у вигляді ієрархії, яка демонструється на рисунку 22.

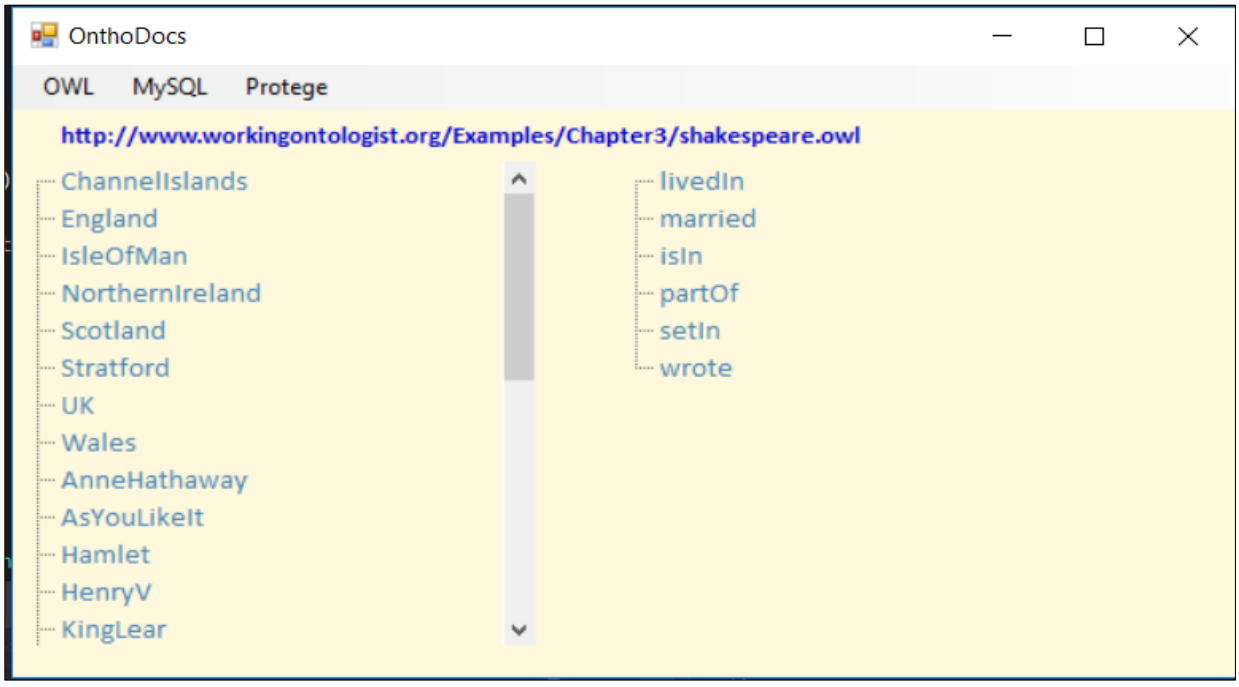

Рисунок 22 — Відображення онтології зі збереженого owl-файлу

Крім цього, користувач може завантажувати онтологію із завантаженого owl-файл до бази даних за допомогою функції “зберегти онтологію у MySql”, що відображено на рисунку 23.

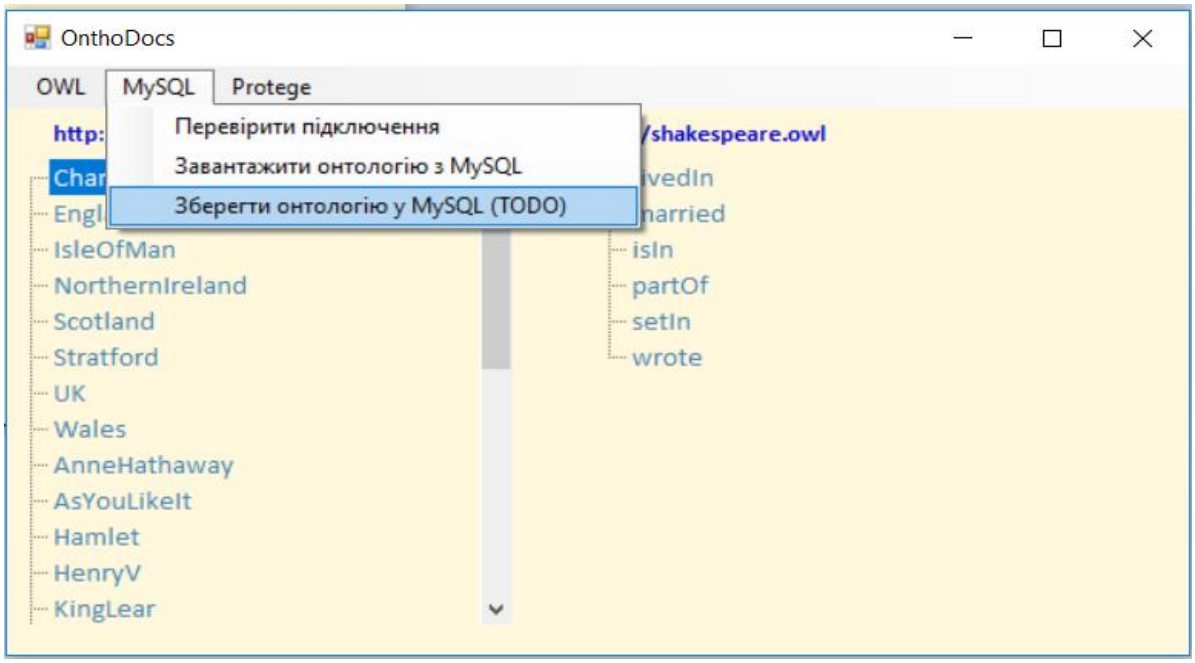

Рисунок 23 — Збереження онтології до бази даних

Також користувач може працювати з онтологією у WebProtege. Зображення опції на рисунку 24.

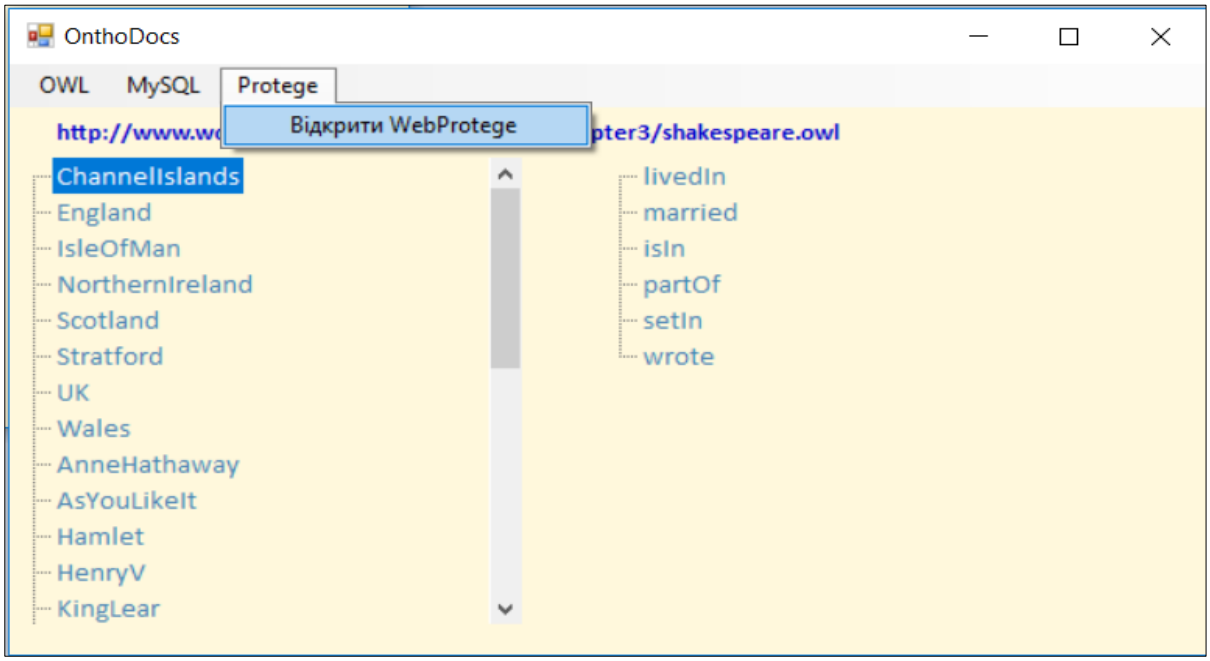

Рисунок 24 — Опція “Відкрити у WebProtege”

При натисненні на кнопку “відкрити WebProtege” у браузері користувача за замовчуванням відкривається вкладка “WebProtege”. Зображено на рисунку 25.

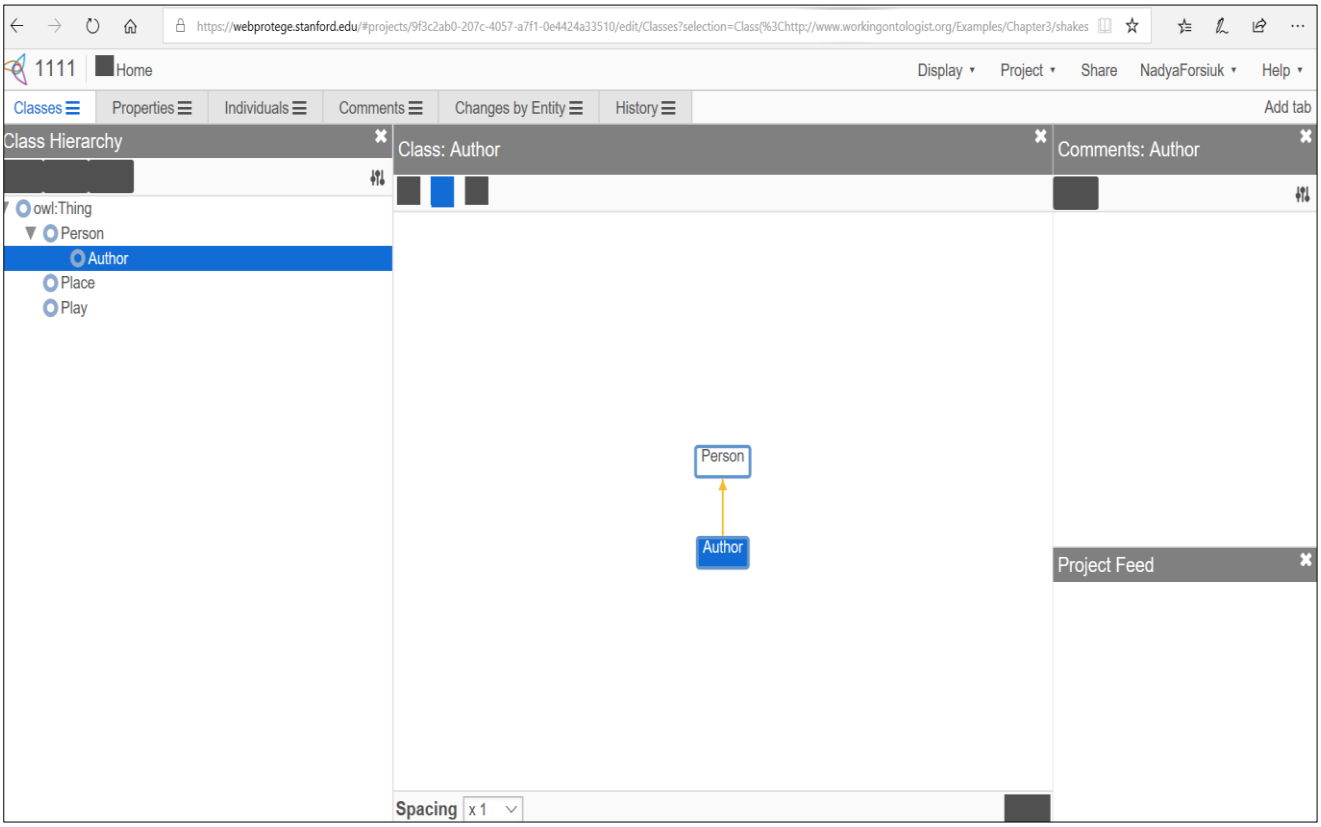

Рисунок 25 — Вкладка WebProtege у браузері користувача

Також, owl-файли структур онтологій, побудованих по заданому метаопису реляційного типу, можна відкрити і працювати з такими файлами у будь-якому редакторі для роботи з онтологіями.

На рисунку 26 зображено приклад роботи з owl-файлом, створеним у додатку OntoDocs, у редакторі Protégé (побудова графу).

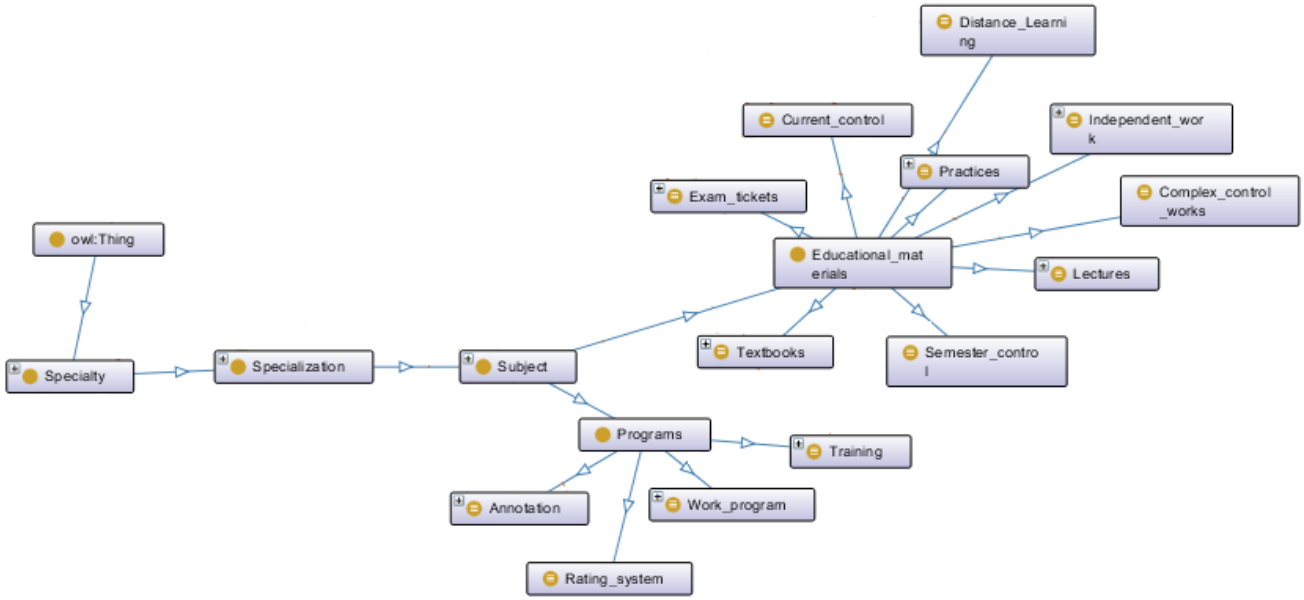

Рисунок 26 – Відкриття owl-файлу, створеного у додатку OnthoDocs, у Protégé

5.1. Висновки до розділу

В розділі описано системні вимоги для запуску програмної системи, загальні інструкції по роботі з нею для кінцевого користувача.

ВИСНОВКИ

У ході виконання дипломної роботи було:

— описано сутність понять «інформаційних ресурсів», «реєстру інформаційних ресурсів» та «онтології»;

— проаналізовано існуючі методи представлення інформації, збереженої у реляційній базі даних у вигляді онтології; існуючі формати її опису та аналоги;

— розглянуто можливість побудови реєстру інформаційних ресурсів з використанням онтологічного підходу та проаналізовано існуючі аналоги;

— запропоновано зберігати реєстр інформаційних ресурсів (метаопис інформаційних ресурсів) в реляційній структурі;

— описані основні програмні засоби розробки, використані для побудови системи, що пропонується, та їх переваги над існуючими аналогами;

— описані технічні деталі реалізації та архітектурні рішення, що були використані при побудові системи;

— побудовано алгоритм розгортання структури онтології з мета бази даних;

— розроблено програмний продукт, який вирішує такий спектр задач:

а) перегляд ієрархічної структури та відношень у складі онтології;

б) відображення онтології з реляційної моделі до OWL-файлу;

— описано системні вимоги для запуску програмної системи, загальні інструкції по роботі з нею для кінцевого користувача.

Система є зручною та простою у застосуванні з інтуїтивно зрозумілим інтерфейсом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Burnett K., Kwong Bor Ng, Park S. A comparison of the two traditions of metadata development /J. of the American Society for Information Science. Special issue on integrating multiple overlapping metadata standards, Vol. 50, Issue 13, 1999, pp.1209-1217. [Електронний ресурс] — Режим доступу: http://comminfo.rutgers.edu/~kbng/publications/1999JASISPark.pdf

2. Башмаков А.И., Башмаков И.А. Механизмы наследования, выявления и разрешения противоречий в обобщенной модели представления предметной области. Часть I //Техническая кибернетика. – 1994, № 5. – С. 14-27.

3. Web Ontology Language, Use Cases and Requirements.W3C Recommendation 10 February 2004. URL: [Електронний ресурс] — Режим доступу: https://www.w3.org/TR/webont-req/OWL

4. Грушков, А.С. Хранилище данных / А.С. Грушков, Е.В. Костюков. – СПб.: СЗИМИ, 2007. – 864 с.

5. Gruber T.R. A translation approach to portable ontologies // Knowledgeition. 1993. – №5(2). – P. 199-220.

6. Guarino N., Guaretta P. Ontology's and Knowledge Bases. Towards a logical Calcification // Towards Very Large Knowledge Bases. – Amsterdam: press, 1995

7. Гаврилова Т.А. Хорошевский В.Ф. Базы знаний интеллектуальны систем. – СПб: Питер, 2000. – 384 с.

8. Гаврилова Т. Онтология для изучения инженерии знаний // Труды Международной научно – практической конференции KDS-2001, 2001.

9. Городецкий В.И., Самойлов В.В., Малов О. А. Современное состояние технологии извлечения знаний из баз и хранилищ данных. Ч.I // Новости искусственного интеллекта. – 2002, № 3. – С. 3-12.

10. Когаловский М.Р. Доступ к реляционным базам данных, основанный на онтологиях /Программирование, МАИК/Наука «Интерпериодика». 2012. № 4.

11. National Information Standards Organization; Rebecca Guenther; Jaqueline Radebaugh. Understanding Metadata. 2004

12. Тузовский А.Ф. Разработка системы управления знаниями на основе единой онтологической модели // Известия Томского политехнического университета. – 2007. – Т. 310. – № 2. – С. 182–185.

13. Taylor C. An Introduction to Metadata. The University of Queensland, Australia.:[Електронний ресурс] — Режим доступу: http://www.library.uq.edu.au/papers/ctmeta4.html

14. [Електронний ресурс] — Режим доступу: http://www.cs.uml.edu/~cchen/ontology/comwim_04.pdf]

ДОДАТОК 1

Web-система для редагування та аналізу генетичних послідовностей

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ51197_19Б

Аркушів 2

Київ – 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ТІ 51197_19Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ТІ 51197_19Б 12-2	Category.cs Document.cs Program.cs Form1.cs DAO.cs	Модулі розробленої системи

ДОДАТОК 2

Клас взаємодії системи та її поточної онтології з базою даних системи MYSQL

Лістинг програмного модулю

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ51197_19Б 12-1

Аркушів 7

Київ – 2019

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MySql.Data.MySqlClient;

namespace Onthology
{
   class DAO
   {
      private Random random = new Random();
      private MySqlConnection connection;
      private String connectionInfo = "Database=onthology;Data Source=localhost;User
Id=root;Password=Krasnozoranuy6b";
      //Інформація для підключення до БД
      public DAO()
      {
         CreateConnection();
         TestConnection();
      }

      private void CreateConnection()
      {
         connection = new MySqlConnection(connectionInfo); //Створення підключення
до БД
      }
      public void TestConnection() // тестування створеного підключення
      {
         connection.Open();
         connection.Close();
      }

      public List<Category> LoadFromDB()    //завантаження інформації з БД
      {
         String query1 = "select * from class";
         MySqlCommand command = new MySqlCommand(query1, connection);
         MySqlDataReader reader = null;
         List<Category> categories = new List<Category>();
         List<Document> documents = new List<Document>();
         try
         {
            connection.Open();
```

```
reader = command.ExecuteReader();
while (reader.Read())
{
   if (!Boolean.Parse(reader[2].ToString()))
   {
      Category category = new Category();
      category.ID = reader[0].ToString();
      category.Name = reader[1].ToString();
      categories.Add(category);
   }
   else
   {
      Document document = new Document();
      document.ID = reader[0].ToString();
      document.Name = reader[1].ToString();
      document.Content = reader[3].ToString();
      documents.Add(document);
   }
}
reader.Close();
connection.Close();
connection.Open();
String query5 = "select * from hierarchy";
command = new MySqlCommand(query5, connection);
MySqlDataReader reader5 = command.ExecuteReader();
while (!reader5.IsClosed && reader5.Read())
{
   String superId = reader5[1].ToString();
   String subId = reader5[2].ToString();
   Document doc = FindDocumentById(documents, subId);
   if(doc != null)
   {
      Category category = FindCategoryById(categories, superId);
      category.Documents.Add(doc);
      doc.Category = category;
   }
   else
   {
      Category superClass = FindCategoryById(categories, superId);
      Category subClass = FindCategoryById(categories, subId);
      superClass.Subcategories.Add(subClass);
      subClass.Supercategories.Add(superClass);
   }
```

```
            }
            connection.Close();

            return categories;
        }
        catch (Exception e)
        {
            System.Windows.Forms.MessageBox.Show(e.Message + "\r\nПомилка під час завантаження даних з бази");
            return null;
        }
        finally
        {
            connection.Close();
        }
    }

    private Category FindCategoryById(List<Category> list, String id)
    {
        foreach(Category category in list)
        {
            if (category.ID.Equals(id))
            {
                return category;
            }
        }
        return null;
    }

    private Document FindDocumentById(List<Document> list, String id)
    {
        foreach (Document document in list)
        {
            if (document.ID.Equals(id))
            {
                return document;
            }
        }
        return null;
    }

    public void SaveToDB(List<Category> categories)
    {
        try
```

```
        {
            String query1 = "delete from class; delete from hierarchy;";
            MySqlCommand command = new MySqlCommand(query1, connection);
            connection.Open();
            command.ExecuteNonQuery();
        }
        catch (Exception e)
        {
            System.Windows.Forms.MessageBox.Show(e.Message);
        }
        finally
        {
            connection.Close();
        }

        String newID = "1";

        foreach (Category category in categories)
        {
            try
            {
                String query2 = "insert into class values (" + newID + ",'" + category.Name +
"','"
                    + "false" + ", " + "null); ";
                MySqlCommand command2 = new MySqlCommand(query2, connection);
                connection.Open();
                command2.ExecuteNonQuery();
                category.ID = newID;
                newID = "" + (Int32.Parse(newID) + 1);
            }
            catch (Exception e)
            {
                System.Windows.Forms.MessageBox.Show(e.Message);
            }
            finally
            {
                connection.Close();
            }
        }

        foreach (Category category in categories)
        {
            if (category.Supercategories.Count == 0)
            {
```

```
                AddRecursively(category);
            }
        }
    }

    private void AddRecursively(Category category)
    {
        int i = 0;
        foreach (Category subcategory in category.Subcategories)
        {
            try
            {

                String query2 = "insert into hierarchy values (" + random.Next(65545) + "," +
category.ID +
                    ", " + subcategory.ID +"); ";
                MySqlCommand command2 = new MySqlCommand(query2, connection);
                connection.Open();
                command2.ExecuteNonQuery();
                AddRecursively(subcategory);
                i++;
            }
            catch (Exception e)
            {
                System.Windows.Forms.MessageBox.Show(e.Message);
            }
            finally
            {
                connection.Close();
            }
        }
    }

  }
}
```

ДОДАТОК 3

Клас взаємодії системи та її поточної онтології з базою даних системи MYSQL

Опис програмного модулю

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ51197_19Б 12-2

Аркушів 6

Київ – 2019

АНОТАЦІЯ

Додаток надає короткий функціональний та структурний опис реалізації класу DAO(Data Access Object), що відповідає за операції взаємодії системи та її поточної онтології з базою даних системи MYSQL.

Екземпляр класу DAO відповідає за операції взаємодії з реляційною базою даних. Методи даного класу викликаються при обробці специфічних подій графічного інтерфейсу.

Клас DAO реалізований за допомогою фреймворку ADO .NET та розширення MySql.Data.

ЗМІСТ

-4-

ЗАГАЛЬНІ ВІДОМОСТІ

Програмний код міститься в межах класу DAO(Data Access Object).

Екземпляр класу DAO відповідає за операції взаємодії з реляційною базою даних.

Методи даного класу викликаються при обробці специфічних подій графічного інтерфейсу.

Програма була написана мовою C# з використанням фреймворку ADO.NET та розширення MySQL.Data.

-5-

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Клас реалізує наступний функціонал:

— створення підключення до бази даних;

— тестування встановленого підключення;

— безпосереднє завантаження онтології з бази даних(із заміщенням поточної онтології);

— збереження поточної онтології до бази даних(із заміщенням онтологічної схеми, яка на той момент існує у базі даних);

-6-

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Модуль реалізовано у формі класу, що реалізує відповідний інтерфейс, що містить оголошення та короткий опис наступних методів:

1) private void CreateConnection();
2) public void TestConnection();
3) public List<Category> LoadFromDB();
4) public void SaveToDb(List<Category>categories);

-7-

ТЕХНІЧНІ ЗАСОБИ, ЩО ВИКОРИСТОВУВАЛИСЯ

Для потреб взаємодії із зовнішніми джерелами збереження даних було використано готові рішення у вигляді фреймворків ADO.NET та dotNetRDF(DotNetOwlAPI).

Для взаємодії системи з онтологіями використовувалося розширення MySQL.Data.